A morphosyntactic processor of modern standard Arabic

Degachi, Abdelmajid

Award date:
1990

Awarding institution:
University of Bath

A MORPHOSYNTACTIC PROCESSOR OF MODERN STANDARD ARABIC

submitted by

Abdelmajid Degachi

for the degree of Ph.D.

of the

University of Bath

December, 1990

School of Modern Languages
and International Studies

Signature of Author .....

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Dedication

To the spirit of Lenore Marshall,

To my ABLA,

To my dear parents and all my family,

To my beloved country, TUNISIA.

Acknowledgements

I would like to express my deep gratitude to all the people who have sponsored me for this degree. I wish especially to thank the British Council and all their staff in Bristol and Tunis for their grants of one year to do an M.A. degree and of four years to do this Ph.D. degree, for their travel grants to attend important conferences on computational linguistics in Lugano, Stanford, and Geneva, as well as for all their help and support. I thank the Overseas Research Students Awards Scheme for their concurrent award of three years for this degree, the Arab-British Chamber Charitable Foundation for an award of £1300, the Africa Educational Trust for an award of £500, and the University of Bath for a travel grant to attend an international conference on computational linguistics in Bonn. I also wish to express my sincere thanks and deep gratitude to my friend Mike Marshall for his personal sponsorship and friendship as well as to Mr J. C. Smith for his moral and financial support as funds were steadily eroding.

Much acknowledgement is also due to Mr J. C. Smith for his advice, suggestions, and support as well as for his diligent supervision and unfailing assiduity in reading the material of this thesis. Special thanks are also due to all the people I work with in the Mathematics Computer Lab, namely to Professor J. Fitch for letting me use the computer facilities and attend his LISP and Logic courses, Professor J. Davenport, Dr R. Bradford, Dr J. Padget, Mike Dewar, and Dr Dave Lavender for their help with LaTeX and other technical problems on the computer. I also thank Dr Abdelmajid Ben Hamadou for giving me copies of his articles, Dr R. Riad and Dr M. Maiden for lending me some books and references, and Professor R. Souissi and Mr Nick Alt for their help. I am also indebted to all my teachers at l'Ecole Normale Supérieure de Tunis: Mrs S. Bouzouita, Mrs S. Jamoussi, Mr G. Stott, Dr B. Lamine, Miss Moalla, Mr A. Chalbi, Mr M. Shaawish, and Dr P. Vernon.

Finally, I wish especially to thank all my family and my friends for their patience, their moral support, and friendship during the difficult course of my studies.

Obviously, any shortcomings in this work are the responsibility of the author and no one else's.

Summary

In this thesis, we deal with practical and theoretical aspects of morphological processing by computer with special reference to Modern Standard Arabic. We give a critical review of previous Arabic and Western approaches to morphological analysis and previous attempts to analyse Arabic morphology by computer and we explain the inadequacy of the previous approaches and analyses. We then propose an alternative linguistic framework and a technique for the analysis of Arabic morphology by computer. The technique, which is based on a theory of Arabic word structure, is faster and more efficient than previous essays in this field.

We start the linguistic analysis by describing formal models for Arabic Verbs, Nominals, and Particles. We analyse their syntactic, morphological, and graphological structures. The models enable us to construct Verb, Nominal, and Particle databases and to implement a V-Processor, a N-Processor, and a P-Processor for the efficient parsing of Arabic Verbs, Nominals, and Particles. We then synthesize our linguistic and computational analyses and integrate the resulting syntheses into a general model that allows us to implement a general morphological processor which includes all the databases, coordinates the work of all the processors, and optimizes the parsing of Arabic Polysynthetic words.

Finally, we evaluate the validity and viability of our linguistic and computational analyses, we assess the possibilities for the implementation of a syntactic parser for Arabic, and we suggest prospects for further research in this field.

Transcription Symbols

a. Consonants and Semivowels

1)	:	.....ء.....	15)	d;	....ض....
2)	b	....ب....	16)	t;	.....ط....
3)	t	....ت....	17)	d/	.....ظ....
4)	t-	....ث....	18)	ɛ	.....ع.....
5)	j	.....ج....	19)	g	.....غ....
6)	h-	.....ح....	20)	f	.....ف...
7)	x	.....خ.....	21)	q	.....ق....
8)	d	.....د.....	22)	k	.....ك....
9)	d-	.....ذ.....	23)	l	.....ل....
10)	r	.....ر.....	24)	m	.....م.....
11)	z	.....ز.....	25)	n	.....ن....
12)	s	....س....	26)	h	.....ه.....
13)	sv	....ش....	27)	w	.....و.....
14)	s;	....ص....	28)	y	.....ي.....
			29)	'	.....ا.....

b. Vowels

		Single			Full
1)	u	.....	4)	u+	.....
2)	a	.....	5)	a+	.....
3)	i	.....	6)	i+	.....
			7)	o	.....°.....(mute).

c. Geminate

1) cc = double consonant, (where c is any consonant).

2) ww = double semivowel.

3) yy = double semivowel.

d. Graphological Description

1) Lunar characters: {: b j h- x ε g f q k m h w y}.

2) Solar characters: {t t- d d- r z s sv s; d; t; d/ l n}.

e. Differences with ALECSO

All the above transcription symbols are the same as those decided by ALECSO in TUNIS, 1981 except for these changes where my symbols are given second:

1)	’	:	5)	d̲	d-	9)	ṭ	t;
2)	t̲	t-	6)	š	sv	10)	Ḍ̶	d/
3)	h̲	h-	7)	ṣ	s;	11)	ᶜ	ε
4)	k̲	x	8)	ḍ	d;	12)	g̲	g

==0==<★★★★★>==0==

VOLUME I

Table of Contents

==0==<*****>==0==

List of Tables

==0==<*****>==0==

List of Algorithms and Figures

==0==<*****>==0==

Abbreviations and Notations

Organizations and Standard Terms

ACL	Association for Computational Linguistics.
ALECSO	Arab League Educational Cultural and Science Organization.
ALPAC	Automatic Language Processing Advisory Committee.

ATN	Augmented Transition Network.
C.A.	Classical Arabic.
CFG	Context-Free Grammar.
CL	Cambridge LISP.
CPU	Central Processing Unit.
CSG	Context-Sensitive Grammar.
DCG	Definite Clause Grammar.
GFs	Grammatical Functions.
IA	Item and Arrangement.
IC	Immediate Constituent.
IM	Interpretive Morphology.
IP	Item and Process.
LM	Lexical Morphology.
MA	Morphological Analysis.
M.S.A.	Modern Standard Arabic.
MT	Machine Translation.
NLP	Natural Language Processing.
PSG	Phrase Structure Grammar.
RAM	Random Access Memory.
SM	Syntactic Morphology.
TG	Transformational Grammar.
TM	Transformational Morphology.
WP	Word and Paradigm.

Categories

AA	Indefinite Adjective.
AD	time ADverb.
AM	Annexed Modifier.
AN	Annexed Nominal.
ANF	Annexed NF.
AP	Affirmative Particle.
Ab.	Abbreviated.
Ab.N	Abbreviated Nominal.
Adv	ADVerb.
B	Subjunctor.
C	Coordinative Particle.
CCF	Complex CF.
CF	Conjugation Form.
CNF	Complex NF.
CPF	Complex PF.
CPs	aCcusative Pronouns.
Ce	Centre.
D	Determiner.
DA	Definite Adjective.
DM	Definite NuMeral.
DN	Definite Nominal.
DNF	Definite NF.
DPs	Demonstrative Pronouns.
Dd.	Doubled.
Df.	Defective.
Df.N	Defective Nominal.
Df.V	Defective Verb.
EP	nEgative Particle.
Ex.	Extended.
Ex.N	Extended Nominal.
F	bound Future Particle.
FCF	Future CF.
FD	Free Future Particle.
FP	Free Particle.
G	GENDER suffix.
GPs	Genitive Pronouns.
HQ	Human QP.
Ho.	Hollow.
ICF	Imperfect CF.
INF	Indefinite NF.
JP	Jussive Particle.
JQ	obJect QP.
K	NGC suffix.
L	VerbaL.
L1	VerbaL1.
L2	VerbaL2.

LD	pLace aDverb.
LOC	tLOCative.
MD	MoDifier.
MM	Indefinite NuMeral.
Mh.	'Mahomuwz'.
N	Nominal.
NC	Nominal Complex.
NF	Nominal Form.
NN	Indefinite Nominal.
P	Prefixed Particle/Preposition.
PC	Particle Complex.
PCF	Perfect CF.
PD	Assertive Particle.
PF	Particle Form.
PGP	P/Pr attached to a GP.
PN	Proper Noun.
PVB	Future VerB.
PVR	Future VR.
PW	Polysynthetic Word.
Pr	Free PReposition.
Q2	Interrogative Particle.
QP	Question Particle.
RCF	Referential CF.
RPs	Relative Pronouns.
SPF	Simple PF.
SPs	Subject Pronouns.
So.	Sound.
So.N	Sound Nominal.
So.V	Sound Verb.
TPs	encliTic Pronouns.
VA	Verb/Adjective.
VB	VerB.
VC	Verb Complex.
VM	Verb/Numeral.
VN	Verb/Noun.
VR	Referential Verb.
WN	NN/AN.
WV	VR/AN.
Wk.	Weak.
Wk.N	Weak Nominal.
XA	AA/MD.
XN	NN/MD.
XP	Accusative/Genitive Pronoun.
c	consonant.
e	semivowel.
pref	Prefix.
suf, S	Suffix.
v	vowel.

Properties

CAT	CATEGORY.
CAT1	initial CATEGORY.
CAT2	alternative CATEGORY.
CMK	CASE MARK.
CMK1	initial CASE MARK.
CMK2	alternative CASE MARK.
DC	DFN, CMK.
DFC	DFN, FLXN, CMK.
DFN	DEFINITENESS.
FLXN	FLEXION.
GDR	GENDER.
GVT	CASE GOVERNMENT.
HTY	HUMANITY.
MDG	MOOD GOVERNMENT.
MMK	MOOD MARK.
MODE	interrogative/declarative.
NBR	NUMBER.
NG	NBR, GDR.
NGC	NBR, GDR, CMK.
NGP	NBR, GDR, PRS.
PLC	PLURAL CLASS.
PRS	PERSON.
REF	REFERENCE.
TNS	TENSE.
TRVY	TRANSITIVITY.
TYP	TYPE OF PLURAL.
VAL	NUMERICAL VALUE.
VOC	VOICE.

Property Values

ACC	DEEP ACCusative CASE.
NEUT	NEUTral.
NOM	DEEP NOMinative CASE.
OBL	DEEP oblique CASE.
acm	ACcusative CASE MARK.
act	ACTive.
agr	AGReement.
bpr	Bound to PReposition.
col	COLlocutive.
cpm	aCm or obm CASE MARK.
ctr	COTRansitive.
dfp	DeFinite.
dl	DuaL.
dnf	reDuced NGC suFfix.
dp	Dual/Plural.
dtr	DiTRansitive.
exl	EXLocutive.
ff	Feminine.
fvc	deFeCtiVe.
fvm	Full Vowel Mark.
hm	of HuMan agr.
hp	of Potential Human agr.
ids	InDicative, Subjunctive.
imp	IMPerfect.

ind	INDicative.
int	INTerrogative.
int	INTerrogative.
jis	Neutral of MOOD.
jus	JUSsive.
mc	Masculine.
mf	neutral of GENDER.
mtr	MonoTRansitive.
mxtr	Monotransitive/XTRAnsitive.
nbd	free, Non-BounD.
ncp	Neutral of CASE.
nctr	Ntr/CTR.
ndf	iNDeFinite.
nmm	NoMinative CASE MARK.
nnf	Full NGC suFfix.
npm	Nmm or obm CASE MARK.
ntr	iNTRansitive.
obm	OBlique CASE MARK.
pas	PASsive.
pl	PluraL.
prf	PeRFect.
ptr	PTRansitive.
ptr1	PTRansitive1.
ptr2	PTRansitive2.
r1	first PERSON.
r2	second PERSON.
r3	third PERSON.
rcm	Restricted to aCm.
rfp	Regular Feminine Pl.
rmp	Regular Masculine Pl.
rmx	rmp or sn.
rnc	Restricted to Nmm and aCm.
sjj	SubJunctive, Jussive.
sn	SiNgular.
sp	Singular/Plural.
sub	SUBjunctive.
svm	Single Vowel Mark.
xdf	of ambiguous DEFINITENESS.
xh	of ambiguous Human agr.
xtr	XTRansitive.
xvm	svm or fVM.
xxl	of ambiguous REFERENCE.

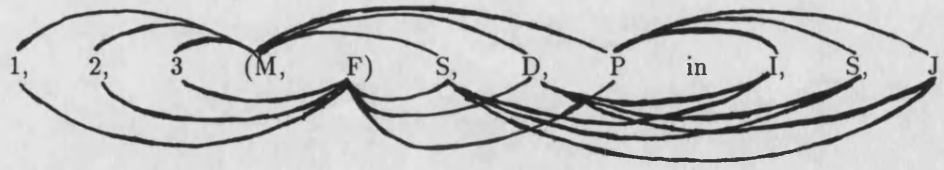
Miscellaneous

$\mathcal{IFF}$	IF and only iF.
@	for All PERSONs.
ADJP	Adjective Pattern.
ASRs	Affix Selection Rules.
App.	APPendix.
BP	Broken Plural.
CRD	Causative Root Dictionary.
DAP	Derivative Act Pattern.
DSP	Derivative paS Pattern.
FBP	Feminine BP.
FO	Feminine Only.
FRP	Feminine RP.
FWT	Function Word Table.
GCs	Graphotactic Conditions.
IRD	Intensive Root Dictionary.
LOC VAR	LOCal VARiable.
LOCVP	Tlocative Pattern.
LR	Left-to-Right.
LRL	Left-to-Right-to-Left.
MBP	Masculine BP.
MO	Masculine Only.
MPACT	iMPerfect ACT.
MPPAS	iMPerfect PAS.
MRP	Masculine RP.
MS	Morphological Structure.
N/A.	Not Applicable.
NNP	(Substantive) Noun Pattern.
NRVB	iRregular VerB roots.
NUMP	Numeral Pattern.
P/R	Pattern/Root combination.
P/S	Pattern/Stem combination.
PRACT	PeRfect ACT.
PRPAS	PeRfect PAS.
RL	Right-to-Left.
RP	Regular Plural.
RRVB	RegulaR VerB roots.
SEC	SECond.
VCo	Vocalic COmpatibility.
Xer	CHAracter.
ans	ANSwer.
b.	Born.
c.	Circa.
cond	CONDition.
d.	Died.
fn.	FootNote.
p.	Page.
pp.	Pages.
res	RESult.
st	Such That.
str	STRucture.

Notations

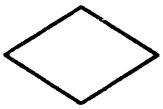
+	obligatory order/condition.	~	free order.
$	boundary, attached to.	∀	forall.
*	ungrammatical.	!	semantically odd.
<	strictly less than.	≤	smaller than or equal to.
>	greater than.	≥	greater than or equal to.
<1	1st Xer of a word.	<2	2nd Xer of a word.
>1	final Xer of a word.	>2	penultimate Xer of a word.
≠	not equal, different.	Ø	null, empty set.
=	equal to.	<x>	Xer/grapheme.
∞	infinite.	≡	equivalent.
( )	optional.	( )*	optional and recursive.
⟶	rewrites/LR direction.	⟵	RL direction.
⇾	becomes, is transformed to.	⟹	means, implies.
/ [ ]	in the context.	/ — [ ]	in the context before.
/ [ ] —	in the context after.	L (x)	LENGTH of x.
f (x)	any function of x.	...	unspecified material.
Ω	agree.	#	disagree.
ØΩ	in neutral agr.	π	govern.
πa	annexation GVT, type1.	πb	annexation GVT, type2.
πc	CASE governs.	πm	MOOD governs.
πs	specification GVT.	[+x]	required feature x.
[−x]	forbidden feature x.	[...]	(in quotations) my comment.
$x = [i, j]$	x lies in the interval i to j inclusive.	{x,y,z}	x, y, and z.
{x/y/z}	x, y or z.	[0 – 9]	positions/order of morphemes, operations.
[a – z]	order of operations.		
%%%	comment.		

1, 2, 3 (M, F) S, D, P in I, S, J. (where 1, 2, 3 are 1st, 2nd, and 3rd PERSON; M, F are mc and ff; S, D, P are sn, dl, and pl; and I, S, J are ind, sub, and jus). This configuration should be read as if it was a distributive multiplication function between the elements of each of its parts. The figure below shows the relations that should be understood:



Flowchart and G-Tree Notation

input/output.



decision.

action/process.

function call.

AND node.

OR node.

==0==<*****>==0==

Table of Errata

Line	Page	Misprint	Correction
last but 4	36	known	known in the literature
9	41	while	Further,
28	51	**First**, The	**First**, the
26	61	*and*	*And*
last	62	Computer	computer
5	69	below).	below.)
26	71	there	There
last but 1	89	phenomena	phenomenon
last but 7	96	:ifoɛalo, :ufoɛulo.	:ifoɛalo, :ufoɛulo, :ifoɛilo.
5	101	PTRANSITIVE2	PTRANSITIVE1, PTRANSITIVE2
7	101	PTRANSITIVE1	PTRANSITIVE
11	113	V-Patterns:	V-Pattern:
7	116	Sin	S in
18	163	Object	Accusative
28	163	Object	Accusative
last but 1	176	Noun	noun
8	196	will grow exponentially	will have multiplicative growth
last but 3	228	lilobawa|wa|niy	lilobawa|niy
10	229	"my teachers"	"(the) teachers (of)"
last but 7	229	limudar risa|tiy	limudarrisa|tiy
24	232	CFG	CSG
20	239	<i+> $	<i+>$
last but 9	289	i:nna	:inna
15	306	Processor	Processors
7	329	(EXSYMBOLS	<EXSYMBOLS
last but 6	330	Figure 52, above,	Figure 52, below,
10	352	$F \quad \pi c \quad V.$	$F \quad \pi m \quad V.$
21	355	$e.i < i.,$	$e.ii.,$

INTRODUCTION

I PRESENTATION AND MOTIVATION OF TOPIC

The research interest of this thesis falls within the field of linguistic analysis by computer. In particular, we are concerned here with the context-free *morphological analysis* (henceforth MA) of Arabic from the linguistic and from the computational points of view; and our aim is to develop a practical and efficient approach to the *computational parsing* of Arabic which is based on a sound theory of *Arabic morphology*. (We mean by *morphology* the study of the different forms of words, their internal structure, and the variation inside this structure.) Thus, the scope of this thesis is the intersection between Arabic morphology and computation.

I.1 THESIS OUTLINE

The introduction presents the topic of this thesis and explains the motivation behind it. It also discusses some definitions, assumptions, limitations, and difficulties, as well as the methodology, terminology, typography, and conventions, of our analysis.

The thesis itself is divided into five parts and twelve chapters. Part One (Chapters 1, 2, and 3) discusses generalities of morphological and computational analysis; Part Two (Chapters 4 and 5) analyses Arabic *Verbs*; Part Three (Chapters 6 and 7) analyses Arabic *Nominals*; Part Four (Chapters 8 and 9) analyses Arabic *Particles*; and Part Five (Chapters 10, 11, and 12) analyses Arabic *Polysynthetic words* and evaluates the analysis.

In Chapters 1 and 2, we deal with theoretical issues in morphology and in *parsing*. We give a critical review of previous Arabic and Western approaches to MA and of previous attempts to analyse Arabic morphology by computer. We then propose an alternative linguistic framework for, and our own computational approach to, the analysis of Arabic morphology. In Chapter 3, we deal with practical issues in computation and programming languages and with pertinent processing generalities and necessary procedures for the operation of our parser.

In Chapters 4, 6, and 8, we define the Arabic Verb, Nominal, and Particle, respectively. We analyse their constituent structure, conjugation, derivation, inflection, features, and functional requirements according to syntactic, morphological, and graphological criteria. We also deal with homonymy, affixation conditions, (V-, N-, and P-) Morphological Structure (henceforth MS) rules, and the construction of (V-, N-, and P-) formal models which enable their efficient parsing.

In Chapters 5, 7, and 9, we translate the descriptions and analyses of Chapters 4, 6, and 8, respectively, into formal specifications and Verb, Nominal, and Particle databases. We implement a V-Processor, a N-Processor, and a P-Processor whose tasks include, respectively, the identification of Verb, Nominal, and Particle structures, the assignment of properties to them, the resolution of ambiguities, and the application of the appropriate linguistic conditions and constraints. We also make linguistic and computational syntheses of the (V-, N, and P-) Processors.

In Chapter 10, we integrate the (V-, N-, and P-) analyses into a formal and general model for Arabic Polysynthetic words. We deal with general affixation conditions, disambiguation, and MS rules which enable the efficient parsing of Arabic words.

In Chapter 11, we convert the definitions of Chapter 10 to formal specifications and a general database. We implement a PW-Processor (called TUNIS1) which coordinates the work, and controls the interactions, of the previous processors. We make a general linguistic and computational synthesis of our analysis.

In Chapter 12, we evaluate the linguistic validity and the computational performance and viability of our approach, especially with respect to the feature system and the anticipation of syntactic applications. We investigate the morphological and syntactic interface and the generation of Arabic sentences. Finally, we assess the prospects for implementing a *syntactic* parser of Arabic.

We conclude by summarizing the practical and theoretical results of the computational and linguistic analyses and suggesting some prospects for further research in this field.

In Volume II, we provide examples of Arabic Verb conjugations, Nominal patterns, and morphological CATEGORIES (in Appendix A), a listing of the databases (in Appendix B), a description of the program (in Appendix C), automatic traces for the program (in Appendix D), an English glossary of Arabic lexical entries (in Appendix E), and an English glossary of Arabic grammar terms (in Appendix F).

I.2 IMPORTANCE OF THEORETICAL MORPHOLOGY

In order to emphasize the importance and complexity of morphology in linguistic study, we note that ACL, 1985: 184, reports that there are "examples of Central and South American languages with up to 15 prefix slots and 28 suffix slots"; while JÄPPINEN and YLILAMMI, 1986: 257, report that, in Finnish, "a nominal may appear in a running text in thousands of different forms, and verbs have an even wider spectrum of forms". However, such facts have come to the forefront of linguistic research only recently.

The 1940s saw a flourishing of morphological studies following BLOOMFIELD, 1933, and the structuralist school which considered morphology as the centre of linguistic analysis. However, since the 1950s and the development of CHOMSKY's generative grammar, the study of morphology has been almost totally neglected. Thus, MATTHEWS, 1974: 4, notes that "if the 1930s were for structural linguistics above all a decade of phonology, and the 1940s and early 1950s a period of apparently parallel progress in morphology, the 1960s in particular have been a decade of syntax ".

Generative grammar itself has especially neglected morphology, a neglect which is recognized by TRAVIS, 1978: 1, as well as by several others. Hence, ARONOFF, 1976: 4, concedes that "within the generative framework, morphology was for a long time quite successfully ignored". Similarly, SELKIRK, 1982: 1, reports that "considerably less attention has been paid to the structure of words" than to syntax. One of the main reasons for this neglect, as noted by MATTHEWS, 1974: 17, is that morphology is "prima facie something of a challenge to the universalist hypothesis"—since, as MATTHEWS argues, ibid. 18, "some [languages] are said to have a very complex morphology; others none at all" and thus, "a set of universals is not to be expected [in morphology] on a priori phonetic or semantic grounds"—whereas we note that the universalist hypothesis has been, at least, a motivation and a goal for most modern linguistic work.

I.3 IMPORTANCE OF ARABIC MORPHOLOGY

Arabic is a Semitic language which is among the first six languages most spoken in the world today (cf. BRUSSET and ABDELGHANI, 1989: 10). Unlike some Indo-European languages, Arabic has a relatively free word order (cf. DEGACHI, 1984: passim) so that much of the grammatical information is supplied at the morphological level. Yet, little attention has been paid to morphology in modern and recent Arabic studies.

This neglect of morphology is a result of direct and naïve copying of linguistic work on languages such as English and French, where the focus has been on syntax, since the structure

of words is relatively simple and not much MA is required. In fact, Arabic is highly inflected with a morphology that is so complex that STETKEVYCH, 1970: 2, was led to exclaim that it

> suggests an idea of almost mathematical abstraction. The perfect system of the three radical consonants, the derived verbal forms with their basic meanings, the precise formation of the verbal noun, of the participles—everything is clarity, logic, system, and abstraction. The language is a mathematical formula.

In the words of SMEATON, 1956: 364, Arabic is characterized by "its predominant use of the discontinuous morpheme, with the result that the identification of the lexeme depends in an unusual degree upon the recognizability of its constituent elements". It was this complexity of the problem of discontinuous morphemes which drove ARONOFF, 1976: 70, to simply brush aside Semitic morphology in his theory, in this brief statement:

> though, I cannot claim to have solved all the mysteries of Semitic morphology, it is clear that once we stop thinking of these vowel patterns as items of the same sort as the stems, we can stop worrying about the metaphysical [sic] import of these discontinuous patterns.

I.4 IMPORTANCE OF COMPUTATIONAL LINGUISTICS

The implementation of linguistic theories in the field of language analysis by computer has largely been neglected. Thus, RITCHIE and THOMPSON, 1984: 359, note that "until about 1965, there was very little overlap between computational work on NLP [Natural Language Processing] and research in theoretical linguistics". The analysis of morphology—despite its importance and complexity—has been especially neglected in computational linguistics. Thus, GAZDAR, 1985: 8, notes that "until recently there has been relatively little computational work on morphology or phonology, in contrast to the large amount of work on syntax and speech". Similarly, KOSKENNIEMI, 1983: 12, opines that "the simplicity of English word inflection has restricted the interest in theoretical research in computational morphology". He argues, ibid. 12, that most early approaches to morphological analysis can be characterized as "stripping off the endings". Such early analysis amounted to just a few crude and sometimes even ad hoc rules which eliminated inflectional variation in word forms and reduced them to basic *stems* that can be stored in a dictionary. (A *stem* is defined by CRYSTAL, 1980: 330–331, as the element of a word, which "may consist solely of a single ROOT MORPHEME [...], or of two root morphemes [...], or of a root morpheme plus a DERIVATIONAL AFFIX".) This method is hardly efficient for Arabic with its wealth of morphological phenomena.

Besides the need for MA, there is also a perceived need for Natural Language Processing (NLP) by computer. This is because NLP is a preliminary but necessary and beneficial step for computational applications of linguistic analyses. The advantage of such computational applications is that, in the words of MASTERMAN, 1979: 164, "by using the machine online, knowledge is made internationally accessible which would not be accessible otherwise". As an example, SAGER, 1979: 5, reports that

> over half the budget of the European Parliament goes into expenditure caused by the necessities of multilingualism. One person out of three working for the European Commission is working full time on tasks designed to overcome the language barrier. In 1977 the Commission alone translated over half a million pages, and another 50,000 were translated by outside contractors;

whilst SLOCUM, 1985: 16, asserts that there is a real shortage of qualified human translators. Moreover, according to LEHMANN, 1978: 160, there are "huge amounts of publication currently produced", especially technological and scientific materials. He draws attention to the perceived need for NLP, pointing out, ibid. 161, that "even punctilious scholars failed to note all the properties of lexical items or grammatical constructions until computers produced the nonsensical output which incomplete descriptions of language yield". SMITH, 1989: 77, also makes this point forcefully, arguing for the benefits of Machine Translation (MT) in providing new insights into language and translation.

I.4.1 Brief History of Computational Applications

Modern computer applications include, inter alia, *Artificial Intelligence*, (defined by WILKS, 1979: 27, as "the use of computational methods for the simulation of distinctively human intellectual behaviour, by means of complex knowledge structures and their manipulation"), *information retrieval, question answering, command systems, data collection*, and *databases*.

However, *MT* is one of the most important applications of the computer. (*MT* is defined by ALPAC, 1966: 19, as going by algorithm, i.e., a predetermined finite set of automatic instructions (cf. DARCY and BOSTON, 1983: 6), "from machine-readable source text to useful target text, without recourse to human translation or editing".)

In order to situate this thesis properly, we first start by outlining a very brief history of MT. The development of work in MT and Artificial Intelligence was the direct result of work in mathematical logic, computational symbol processing, cybernetics and self-organizing systems, and the developments of computing machines; as well as the result of the needs for information acquisition and dissemination. The field of MT was very active until the American government commissioned ALPAC to evaluate MT activities. ALPAC, 1966: 19, made a harsh

criticism of the field and concluded that "there has been no machine translation of general scientific text, and none is in immediate prospect". This failure was due to the overambition and overconfidence which pervaded early work. For instance, SLOCUM, 1985: 1, reports that the early researchers had visions of "high-speed, high quality translation of arbitrary texts" and HUTCHINS, 1986: 150, notes that the decade between 1954 and 1966 "may be characterised as one of initial widespread optimism of 'imminent' success".

The ALPAC criticism was immediately followed by a period of disillusionment and regress until around 1975. Then, there was a resurgence of interest in MT (SLOCUM, 1985: 1–2, reports that, in 1984, half a million pages of text were translated by machine). However, the enthusiasm of the early period was replaced with a more realistic and practical approach, which led to the development of new techniques such as *Machine-Assisted Translation* with *pre-editing* (the modification of text before inputting it to a computer) or with *post-editing* (the correction of Machine Translated text after being output from a computer). Hence, BARR and FEIGENBAUM, 1981: 238, note that "in some applications it is worthwhile to have even a very bad translation if it can be done by a computer in a much shorter time (or much more cheaply) than by humans" and WINOGRAD, 1984: 92, asserts that "although fully automatic, high-quality machine translation is not feasible, software is available to facilitate translation. One example is the computerization of translation aids such as dictionaries and phrase books".

I.4.2 The Computational Analysis of Arabic, an Important but Neglected Field

Despite the developments (described above) of NLP in the West, there is a dearth of work on the computational analysis of Arabic in the light of current linguistic theories while the little work that there is in this field has taken place in the context of outmoded machine translation systems, such as WEIDNER and SYSTRAN. Thus, CHAIB, 1982: 33, found that, on the Arabic language, "computational work is still very limited"; ALMISRIYY, 1983: 156, reports that "the use of computers in linguistic study, and particularly in Machine Translation, is not a road that has been followed by Arab linguists". In Machine Translation, for instance, HUTCHINS, 1986: 333, notes that "there is still a relative neglect of Arabic"; and VAUQUOIS, 1987: 188, remarks that "as for Arabic, only a few attempts have been developed" and that "most of the work remains to be performed".

In the most useful survey to date (carried out for ALECSO) on computational Arabic work in the Arab world, ALJAMℰIYYAT, 1985: 5, notes the importance of using computers for technology transfer, especially with

> the growing gap between the developed and the developing countries including the Arab countries, the great need for the Arabization of modern sciences, as well as

the necessity for making such sciences available to researchers.

However, in the 1985 ALECSO study (pp. 21–35), and in another study by ALECSO, 1989: 32–38, it is shown that out of 17 Arab countries questioned, only five countries reported active projects focusing mainly on database work. These numbered seven: 1) Two projects for MT in Morocco and Saudi Arabia; and 2) Five projects for database work and dictionary compilation in Morocco, Egypt, Jordan, Tunisia, and Saudi Arabia. The 1985 ALECSO study also reveals that none of the projects was completed at that time and since no report of further progress is made by ALECSO, 1989, we can only assume that the projects are still at best incomplete. Further, all the institutions that were questioned, according to ALJAMEIYYAT, 1985: 23, 24, and 26, ALECSO, 1989: 32–41, reported difficulties relating to the lack of expertise, of financial investment, of coordination between the Arab countries, and of machine-code standardization.

The focus on database work in the early period of Arabic computational work, also led to an emphasis on statistical methods of analysis and frequency studies. For example, ALLARD, 1970: 485, describes a project to prepare "an Arabic-Latin and Latin-Arabic lexicon needed for the publication of *De Anima*, by Avicenna"; and VAN RIET, 1970: 33, describes a project to produce a statistical analysis for Modern Arabic texts and frequency tables for lexemes, i.e., a lexicon of Arabic words, with a view to a formulation of a "Basic Arabic".

Moreover and in parallel to the early period of the history of MT in the West, computational work in Arabic has started on the wrong footing, with euphoric outlooks and fantastic claims characterizing some of the literature. Thus, ALEAQABIY, 1984: 90, asserts with elation that one can consider "the success [sic] in performance achieved by the machine [i.e., the computer] as evidence for the soundness [sic] of the hypotheses which this performance has been built upon in achieving that great success".

Some of the main reasons for the poverty of the field of Arabic computational analysis are the difficulties encountered by the computational linguist in finding a computer terminal with the facilities to understand and manipulate the peculiarities of the Arabic script. Computerizing Arabic script is no trivial task, especially with the diacritical marks (or 'svakol') and the changing forms of certain Arabic letters (e.g., <ε> or <g>, are written in four different ways, according to context). Hence, FENWICK, 1987: 39, remarks that "displaying or printing English and Arabic with reasonable quality is beyond the capability of simple variants of '188-character' equipment", the most common type of machine in use. Indeed, a plethora of works tackling these difficulties still dominate the scene, today after some forty years have gone by since the first computers appeared. (For e.g., ABBOUD, 1971: 20 et passim, describes a Computer-Assisted Instruction program "to teach American (or English speaking) adult students to write Arabic correctly"; MACKAY, 1976 and 1977, and HISHMAT, 1976, describe attempts to develop an Arabic typesetting system on a computer terminal; HYDER and RICHER, 1977, and HYDER and

IBRAHIM, 1980, deal with the problems encountered in a theory for, and the design of, a system for printing and communication in Arabic-Farsi-Urdu languages; and BECKER, 1984 and 1987, tackles the thorny area of Arabic word processing.)

In addition, there are a few other technical projects, such as those of GHANDOUR, 1976, and HALL and HUSSAIN, 1978, who tackle the problems of developing Arabic software and programming-language interfaces, as well as CALL (Computer-Assisted Language Learning) projects, which were the subject of such conferences as The First National Symposium on Language Teaching, held in Kuwait, 1985 (cf. SAEED, 1985) and The First and Second Symposia on Arabic by Computer, held at the University of Leeds, 1986 and 1987 (cf. BROCKETT, 1986, and SHIVTIEL, 1989, for details of the project to teach Arabic using the BBC micro-computer to Arabic language learners of various levels).

We conclude that there is a perceived need for further efforts in Arabic computerization and especially in Arabic computational linguistics. Therefore, this thesis is particularly timely.

II SOME DEFINITIONS, ASSUMPTIONS, AND LIMITATIONS

We have so far used the term *Arabic* although there are varieties of Arabic (variously characterized as *literary, classical, standard,* and *written*) and there is often controversy and confusion surrounding the use of such terms, since very few studies provide a precise definition for them. However, a few works (to which the reader is referred for further discussion) deal with this issue: PETRÁČEK, 1960: 29–38, BORRMANS, 1961: 363, BLAU, 1973: 173–231, and 1976: 158–190, and ALSAMARRAAIY, 1982: 51–71 and 139–174, all provide extensive remarks and observations which contrast Modern Standard Arabic (hereinafter M.S.A., or simply Arabic), Classical Arabic (hereinafter C.A.), and other varieties. Their remarks cover several areas of style, vocabulary, grammar rules, and syntax.

In this thesis and following WEHR, 1980: vii, and DEGACHI, 1984: 13, we shall use the term *M.S.A.* to refer to the variety of the Arabic language which, throughout the Arab world from Iraq to Morocco, is found in the prose of books, newspapers, periodicals, and letters. This form is also employed by administrative bodies, in formal public address, over radio and television, in religious ceremonial, in national and international contexts such as diplomacy, and in business and political organisations. We use the term *Classical Arabic,* after DEGACHI, 1984: 13, to refer to "the language of the Quran, religious and cultural heritage and classical literature". In addition, C.A. is distinguished by morphological processes which have now become mostly archaic or even absent in M.S.A., such as the *definitization, dualization,* and *pluralization* of Proper Nouns, as defined and illustrated in Chapter 6 of this thesis. Moreover, the following considerations apply throughout this thesis:

1) Phonological, syntactic, and semantic levels of analysis are not specifically dealt with but may be mentioned where directly relevant to the discussion. Thus, we use the term *morphosyntactic* in the title of this thesis, to refer to the morphological level of analysis (as defined above) and the morphosyntactic domains which deal with *linguistic features,* (as listed in Ch. 1, § II.3.1); no other kind of syntactic analysis or theory is implied. Further, we do not deal with idioms, compounding, collocation, and loan words.

2) The duality of the two linguistic and computational levels of analysis and the concomitant dichotomy of the discussion material represent one of the main difficulties in this thesis which has become somewhat lengthy as a result.

3) Whenever an item is used with a special sense, it is defined. Moreover, we shall assume the defined list of abbreviations and notations and the appendices including the Arabic/English glossaries to be an integral part of this thesis. For greater convenience and due to the extensive number of abbreviations needed for our analysis, we also repeat them, wherever possible, in the text of the thesis.

4) We assume a transcription system for Arabic which is modified from ALECSO's 1981 transcription as given at the beginning of this thesis. In this respect, we stress that each of the symbols in the set: { t- h- d- sv s; d; t; d/ } indicates a single character and not two characters. The addition of the distinctive symbols: {- ; / v } to the right side of the letters, rather than the conventional method of typing them above or below the letters, is due to constraints imposed by the machine used for the computational work. However, with English words, the symbol "-" is used as an ordinary hyphen. Similarly, we use the symbol <:> to represent the 'hamoza', or "glottal stop" in Arabic words, but we use it as an ordinary colon with English words.

5) We assume that the reader has a minimum basic knowledge of Arabic and linguistics (and little knowledge of computing, given that this is primarily a linguistics thesis). However, in most respects the thesis is self-explanatory.

6) In addition to the works reviewed in this thesis, we are aware of other material, which due to limited publications and other logistic difficulties, was, unfortunately, not accessible to us. This material includes HEGAZI and SHARKAWI, 1985; THALOUTH and ALDANNAN, 1985; as well as the following works: 1) The Proceedings of the 10th NCC Conference, King Abdul-Aziz University, Jeddah, Saudi Arabia, 1988; 2) The Proceedings of the 2nd Conference on Arabic Computational Linguistics, Kuwait, 27–29 November, 1989; and 3) The Proceedings of the Workshop on Computer Processing of the Arabic Language, Volumes I and II, Kuwait, 14–16 April, 1985. In addition, SCHABERT, 1973, has a German article, of which we could not obtain a readable translation.

III METHODOLOGY

III.1 METHODOLOGY OF PRESENTATION AND DIFFICULTIES OF THE ANALYSIS

Since the work described here is new for the most part, we have had to introduce our own methodological and typographical conventions, which in most cases and of necessity represent a departure from previous work. Obviously, we depart from our own conventions and definitions only when quoting others.

1) Our own translation is given for quotations from non-English (Arabic and French) sources. In quoting traditional Arab grammarians, we have provided, in parentheses, biographical information about them in order to situate their works in time for the benefit of the reader who is not familiar with them.

2) Since this thesis deals with Arabic words and not sentences, the English translations provided for them are necessarily general and out of context. However, Appendix E provides a glossary with more extensive translations into English for all the Arabic words and roots included in the dictionary of our computer program. For all the roots, (listed in the same order as in Ch. 4, § II.1.3), the meanings given in Appendix E, depend on and are restricted to the TRANSITIVITY values and the patterns associated with these roots, as specified in Chapter 4, § II. In addition, in compiling Appendix E, the following sources were used: LA COMMISSION PERMANENTE DE L'ARABE FONCTIONNEL, 1976; IBNU MANDHUWR (1232–1311), 1966; ALBUSTAANIY (1819–1883), 1977; DAR ALMASHRIQ, 1968 and 1986; MAJMAƐ ALLUGAT ALƐARABIYYAT, 1980; and WEHR, 1980.

3) In Chapters 4, 6, and 8, § I.1, we aim to introduce general background to the linguistic material, or data, which is analysed here, as well as the terminology which is to be used in the discussion. The above Sections also allow us to define very precisely the scope of the more formal analysis following them. The Arabic grammatical information given in the Sections is based, in a very broad sense, on the grammar textbooks of ALMAXZUWMIY, 1966; ALSAYYID, 1982; AL-DAHDAAH, 1981; ALRAAJIH-IY, 1983; BAKIYR et al., 1974, 1975, 1977, and 1983; BLACHERE, 1976; HASSAN, 1978: Vol. IV; DHAYF, 1979; NIƐMA, 1973: Volumes I and II; SHARIYF, 1979; and YAƐQUWB, 1986. Such information covers the rules of Arabic grammar, such as the use of specific conjugational and derivational patterns with specific roots, and was extensively modified for a different analysis, classification, synthesis, and presentation in order to suit the purposes of this thesis. This grammatical information has been divided into three parts: the Verb ('fiƐol'), the Nominal (':isom'), and the Particle ('h-arof'): this is the traditional tripartite division of speech in Arabic. We are aware of the existence of a notable study by

ALSAAQIY, 1977: 23–26 and 173–178, which criticizes the tripartite division and postulates the division of Arabic speech into seven parts, (a position which is widely but mistakenly attributed to HASSAAN, 1979, as in KHARMA, 1979: 283). The criteria for ALSAAQIY's hypothesis are explained on a case-by-case basis for each of his word classes and there is not, in his work, a coherent generalization which permits his hypothesis to be independently motivated. However, the syntactic categorization of Arabic words not being a main concern for us here, we have used the tripartite division which has proved to be convenient for the organization of the data into separate chapters. Nevertheless, our descriptions depart from traditional prescriptive styles and provide a more critical and descriptive account of the data on Arabic morphology.

4) Each procedure in our computer program will be described and then broadly outlined in a *flowchart diagram* representing only the significant details of the procedure, whose full details will be specified in a step-by-step scheme, or algorithm. Schemes 1–6 of the program, which represent general and short procedures, are included in the text (cf. Ch. 3), while Schemes 7–36, which are too long to include in the text, are listed in Volume II, Appendix C. In Appendix C, we use an indentation which reflects the logical organization of our program: every function that is embedded in another one is also indented inside it and this embedding applies recursively; and we also use letters together with square brackets as delineators to delimit material enclosed in the brackets. Thus, $aa = [\ x\ y\ z]$ is a notation where x, y, and z indicate computer operations starting with x and ending with z; aa can then be used as an abbreviation for the operations x, y, and z. Integers in the names of functions (in App. C and in the text) indicate a historical value but are of no further significance. The same applies to names of lists.

III.2 TERMINOLOGY

1) In this thesis, I have based myself on the works of CRYSTAL, 1980, MATTHEWS, 1974, and ALLERTON, 1979, as points of departure for the general definition of linguistic terms, on ALJURJAANIY (1339–1413), 1978, and YAʕQUWB, 1986, for the definition of Arabic linguistic terms, on DARCY and BOSTON, 1983, for the definition of computer terms, and on ALI, 1988: 551–569, ALBAKKUWSH et al., 1989, ALKHUWLIY, 1982, CACHIA, 1973, and WRIGHT, 1896–1898, for the translation into English of Arabic linguistic and grammatical terms.

2) It is often difficult to find adequate English translations for the Arabic grammatical and linguistic terminology. Existing terms are often unsatisfactory and at variance with each other. Therefore, we sometimes had to coin completely new terms or define a new usage for familiar terms. A list of the coinages introduced and existing terms used with a different meaning in this thesis includes the following terms: *Bi-Augmented, Mono-Augmented, Tri-Augmented, Poly-Augmented, Augmentative, Polysynthetic, FLEXION, MODE, Hyper Plural, Super Plural, Nominal, Verbal, Verbal1, Verbal2, Instrumental, Exaggerative, Noun of Instance, M-Infinitive,*

collocutive, exlocutive, definitize, futurize, feminize, dualize, pluralize, sylleme, inflector, governor, jussor, indicator, subjunctor, frozen, plurality, temporary indeclinability, regularization, idiosyncratic, and *graphotactic.*

3) We introduce the expression *attach with* in talking about affixing an affix to a free morpheme, but with the affixation starting from the free morpheme: since, in some cases, described in the thesis, the free morpheme is recognized before its affixes, and since we cannot say of such free morphemes that they are *affixed to* other morphemes.

4) We use the terms *perfect* and *imperfect* for past and present, respectively—after WRIGHT, 1896: Vol. I, 55—although the terms *perfective* and *imperfective* are sometimes used instead. We also use the term *human* to translate the Arabic 'ɛa|qil', literally, "rational, endowed with reason".

5) A complete English-Arabic glossary for grammatical terms and for my coinages is provided, for guidance, in Volume II, Appendix F.

III.3 TYPOGRAPHY AND OTHER CONVENTIONS

1) In general, there is an important distinction between items in upper case, those in lower case, and those with initial capital letters. Linguistic property names, such as *NUMBER, GENDER,* and *PERSON,* as well as function names in *Cambridge LISP* (cf. Ch. 3, § I) are all typed in upper case throughout the text in order to differentiate them from ordinary terms with the same name. For example, the term *set* indicates a collection and *SET* indicates a function name. This distinction of ordinary meaning and function name applies to many items such as *list, deletion, substitution, matching,* etc. Further, a term that has one part in upper case and another in lower case, indicates that the first part is a LISP function name, as in *DERIVation, EXPLODed,* and *MATCHes,* where *DERIV, EXPLODE,* and *MATCH* are function names.

2) In order to emphasize the distinction between Arabic grammar terms and English ones, terms with initial capital letters are used to denote category names in Arabic grammar. For instance, what is meant by an Arabic *'mas;odar'*, "Infinitive", is different from what is meant by an English *infinitive.* Terms in lower case are used with their ordinary every-day non-technical meaning. For instance, the term *category* indicates a class of objects and *CATEGORY* indicates a grammatical property as used in this thesis; the term *number* indicates an integer such as 1, 2, etc., and *NUMBER* indicates a linguistic feature with values such as singular, plural, etc. A *Subject* or an *Object* are grammatical categories, while a *subject* is a topic, an *object* is an item, etc.

3) Following conventions in computational work, words which indicate a single item in a program are typed as a single unit, even though they may ordinarily be written as two or more words.

Hence, our spelling of *goto, inputword(s), outputword(s), forall, rootlist,* etc.

4) In this thesis, we make a distinction between symbols with spaces in them and symbols without spaces. Thus, < 1 are two symbols meaning strictly less than 1 and <1 is one symbol referring to the initial character (of a word); > 1 are two symbols meaning greater than 1 and >1 is one symbol meaning final character; < 2 are two symbols meaning strictly less than 2 and <2 is one symbol meaning second character; and so on. Similarly, P is an abbreviation for a prefixed Particle or Preposition but P/R is one symbol indicating a pattern/root combination.

5) In the case of quotations from Arabic sources, the text is reproduced as near to the typography of the original as possible. In all quotations, any insertions or comments added by the author of this thesis are enclosed in square brackets.

6) Authors are quoted in this format: author's SURNAME, YEAR: (Volume (in Capital Roman Numbers) or Appendix), page(s), (footnote).

7) Non-English words are enclosed in single quotation marks.

8) Double quotes are used to indicate either the translation of Arabic words or short quotations from references.

9) Either opening or closing inverted commas are used with elements of the program described here and these elements are Cambridge LISP objects which are described in Chapter 3.

10) The symbols * and $ are the symbols of ungrammaticality and concatenation, respectively.

11) Italics are used to signal the introduction of new terms being mentioned for the first time or being defined and also to indicate words which are being quoted. Underlined items indicate emphasis.

==0==<*****>==0==

Part I

THEORETICAL AND COMPUTATIONAL PERSPECTIVES

Chapter 1

THEORETICAL PERSPECTIVES IN MORPHOLOGY

INTRODUCTION

In this Chapter, we deal with some theoretical perspectives in morphology. We start, in Section I, by reviewing previous theories and approaches to MA. We review the traditional Arab grammarians' approach and Western approaches to MA (insofar as they are relevant to, and can inform, our analysis of Arabic morphology). We end Section I with a critical synthesis of the previous approaches to MA which explains their inadequacy for treating Arabic morphology.

The critical synthesis provides the background to an alternative linguistic framework which we propose, in Section II, for the MA of Arabic. The model we propose involves presenting an adequate account of inflectional and derivational levels in Arabic morphology and the issue of a derivational source for derived Arabic words. The model also involves the specification of a formal morphological grammar and the definition of its terms and structures. The proposed grammar provides the basis for a theory of Arabic word structure.

I THEORIES AND APPROACHES TO MORPHOLOGICAL ANALYSIS

I.1 THE TRADITIONAL ARAB GRAMMARIANS' APPROACH TO MA

Traditional approaches tend to be dismissed as irrelevant and outdated. However, CRYSTAL, 1971: 39, warns that "in contrasting a new approach with an old, it is all too easy to paint a picture in black and white, whereas the reality of the situation is many shades of grey". Thus, PETERSON, 1972: 513, in fact praises the traditional Arab grammarians for having done "a great deal of useful and insightful work in describing and explaining their language" and which he describes, loc. cit., as "abstract and couched in terms which remind us of modern work". Nevertheless, the MA of Arabic by Arab grammarians can be criticized on several counts.

The traditional approach to MA divides morphology between

syntax (which handles inflection in terms of *GOVERNMENT* and agreement (in this thesis, and basing ourselves on a modified definition of the term '*ɛa|mil*', "governor", by ALJURJAANIY, 1978: 150, we use the term *GOVERNMENT* to refer to "the necessity of assigning a particular inflectional ending to a given word"));

morphology (which handles word-formation, the derivation and conjugation of Verbs and the processes of formation for singular, dual, plural, and other types of Nominal); and

phonology (which handles phonetic changes, declinability in Nominals, and consonantal, vocalic, and semivocalic structure and change).

This led to the fragmentation of the treatment of Arabic morphological data between the domains of syntax, word-formation, and phonology. Thus, ALJURJAANIY, 1978: 91, defines '*:alttas;oriyf*', "morphology", by excluding inflection from its domain, as follows: "it is the study of the principles which determine changes in the forms of a given word which are not inflectional changes". However, he defines '*:alnnah-ow*', "syntax", ibid. 259, as

> the study of inflection, indeclinability, and other laws which affect the forms of Arabic structures. It is also the study of the forms of words with respect to ':iɛola|l', "mutation", [including 'qalob', "metathesis", 'h-ad-of', "elision", and ':iboda|l', "substitution"] and it is the study of the principles that determine the grammaticality or otherwise of utterances.

Similarly, SIYBAWAYHI (from the ALBASRA school, d. c.796), 1970, Vol. I, incorporates his analysis of morphology into the analysis of syntax. Thus, his discussion of inflection, or the CASE and MOOD terminations of words, takes place within the framework of syntactic phenomena such as GOVERNMENT, TRANSITIVITY, and word order.

IBNU JINNIY (942–1002), c.1913: 43, 61, 87, et passim, and SIYBAWAYHI, 1970: Vol. II, 339–

340, 428, and 442, present a view of morphology wherein the formation of words (e.g., Nominals and Verbs) is seen as a pattern-based process, whose derivational source is the word, which is the Infinitive (cf. SIYBAWAYHI, 1970: Vol. I, 1, and IBNU JINNIY, c.1913: 63 and 79, as cited below). The output of the derivation undergoes morphological augmentations and changes (e.g., semivocalic modifications, substitution, assimilation (':idoga|m'), and elision (cf. SIYBAWAYHI, ibid. 393–396, 418, 426, 447, 452–476, and IBNU JINNIY, ibid. 58–60, 65–76)). The changes are motivated by phonetic rules (e.g., voicing and pronunciation (cf. IBNU JINNIY, c.1913: 42, 61, 64–70, and SIYBAWAYHI, ibid. 294–297, 303–317, and 344–365)). For example, SIYBAWAYHI, 1970: Vol. II, 123, analyses 'zinat', "weighing", as derived from 'wazana', "to weigh", which conforms to the pattern 'faʿala' and therefore, he decides that 'zinat' was originally 'wazinat' but the initial elements of the pattern, i.e., 'wa', have been deleted.

I.2 EARLY APPROACHES TO MA

I.2.1 The Item and Arrangement Model

Item and Arrangement (IA) is a model of MA which, for MATTHEWS, 1974: 226, "took the morpheme as its basic unit". In IA, the *morpheme* is defined as a minimum meaningful unit which has a linear structure (so, the word "feet" may be represented as FOOT + PLURAL). Further, the morphology defines rules which in turn specify the alternants, or allomorphs, that each morpheme may have.

Since, the aim of IA is to segment words into morphemes, it was forced to propose artificial solutions where it failed to segment words neatly. Such solutions include the *zero morpheme, the replacive morph,* and *allomorphy*, which were suggested to explain the absence of a past morpheme "ed" in examples like "took" from "take". In addition to artificiality, such solutions fail to show connections between related forms, such as "take" and "took". For Arabic, such an analysis would be extremely inadequate. Consider the problems of Broken Plurals: how to segment them and how to relate them to singular forms.

I.2.2 The Item and Process Model

Item and Process (IP) is a model of MA where morphemes are seen as items which are processed according to a number of operations such as *replacement* and *subtraction* (cf. ALLERTON, 1979: 223). IP offers a dynamic and more coherent approach with three stages where one form is derived from another:

1) Base form: e.g., "take".

2) Rule: "take" + "past tense morpheme" {d} $\Longrightarrow$ "took".

3) Process: {d} $\Longrightarrow$ /-u-/.

However, the main problem with IP remains its suggestion that a given form is basic while others are derived from it, which begs the question which form is to be the basic one? For any language, including Arabic, this is a major difficulty. Thus, ALLERTON, loc. cit., criticizes IP, arguing that "the disadvantages of IP model are that it presents data in an apparently historical account, and that it sometimes requires arbitrary choices about which of two forms is basic and which derived"; and MATTHEWS, 1974: 42, asks "can one always say that a lexeme *a* is synchronically prior to a lexeme *b*?". Moreover, ALLERTON, 1979: 224, adds that "the difficulty of the IP model is precisely the difficulty of unrestricted rewrite-rule grammar" in that there are no restrictions on the kind of processes used.

I.2.3 The Word and Paradigm Model

Word and Paradigm (WP) is a model of MA which, for ALLERTON, 1979: 224, "views the word as a more fundamental unit than the morpheme". The difficulty with WP analysis is that in its effort to avoid the problems of segmentation in IA and the problems of processing in IP, it just fell into merely listing all the different forms, such as Verb conjugations, into *paradigms* and *tables* (e.g., "take", "takes", "took", "taken"), and thus failed to capture any generalizations about the distribution of such forms. Hence, ALLERTON, loc. cit., notes that the proponents of WP are "content to specify the phonetic form of a word alongside its lexical meaning and grammatical characteristics, making clear which parts of the total phonological segment realize which categories". For Arabic, where Verbs and Nominals will take hundreds of different forms, the WP approach would be highly uneconomical and inefficient.

For instance, TRAVIS, 1978: 6, 7, and 72, suggests that the morphology of Arabic be analysed within a WP model. However, this WP model merely allows her to list various classes of Verbs and Nominals into paradigms and tables and to list irregularities in the lexicon (p. 9), but does not provide her with means to capture any significant generalizations about the inflectional processes of Arabic word-formation, and especially about interdependencies between non-adjacent morphemes.

I.3 MODERN APPROACHES TO MA

I.3.1 The Transformationalist Approach

The transformationalist approach to MA, which we will dub *Transformational Morphology* (henceforth TM) is mainly due to CHOMSKY, 1957 and 1965. CHOMSKY, 1957: 107, specifies a *transformational grammar* (TG) as comprising a sequence of phrase structure rules, and a sequence of "morphophonemic rules that convert strings of morphemes into strings of phonemes", with "a sequence of transformational rules" connecting the previous two sets of rules. The definition of a grammar is even more striking in CHOMSKY, 1965: 16, where he states "the three major components of a generative grammar: the syntactic, phonological, and semantic components" with no mention of a morphological component. Therefore, the incorporation of morphology into the syntax and phonology placed the onus of MA on the transformational component and necessitated the introduction of many artificial abstractions and complex generative *transformations*; (a generative *transformation* is defined by CRYSTAL, 1980: 362, as "a FORMAL LINGUISTIC operation which enables two levels of structural REPRESENTATION to be placed in correspondence". A transformational rule applies, to each structural description defined in its domain, a structural change which consists in one or more operations, such as movement, insertion, and deletion. For an extensive discussion on transformations, the reader is referred to CRYSTAL, ibid. 362–364). An example of a morphological transformation is the rule of Affix-Hopping for English (cf. CHOMSKY, 1957: 39). This meant that TM resulted in an unnecessary inflation of the *size* of the dictionary, which was filled with surplus, or idiosyncratic, phonological, morphological, syntactic, and semantic data which was necessary for the operation of the generative transformations. McCARTHY, 1981: 405, criticizes TM, pointing out that "morphological transformations potentially allow any arbitrary operation on a segmental string", a position which is also expressed by HUDSON, 1986: 119. Similarly, SCALISE, 1984: 14, concludes that TM "had a number of inadequacies" and was "no longer tenable"; while SELKIRK, 1982: 69, rejects the argument that inflectional morphology is introduced by syntactic transformations. We will see below how such criticisms have led to a substantial revision of the transformational component in TG.

Before that, we comment on an example of arbitrary analysis, which developed as a side effect of CHOMSKY's 1957 and 1965 view of grammar. This is a widespread phenomenon in Arabic linguistics which we might dub *Syntactic Morphology* (SM). SM consists in performing arbitrary morphological segmentation, at the syntactic level, of words (which are normally a single unit in writing) into two or more morphemes that can then be represented as nodes of a higher syntactic constituent. There were examples of SM in early English linguistics, as in CHOMSKY, 1957: 111; however, SM is much more widespread in descriptions of Arabic. For instance, several examples of SM can be found in LEWKOWICZ, 1967: 242 et passim; SNOW, 1965: 28 et passim; KILLEAN, 1966: 47 et passim; ALKHUWLIY, 1979: 32 et pas-

sim; FAAXUWRIY, 1980: 58 et passim; ELTIKAINA, 1982: 69 et passim; TRAVIS, 1978: 135; ALFEHRI, 1982: 14, and 1985: Vol. I, 82; and BAKALLA, 1979: 13, 14 and 16; as well as several others. They all analyse similar Noun Phrases (all of which have the structure Determiner followed by Noun, but with different terminal nodes, or words, for the Noun). For instance, SM analyses the Arabic word ':alowaladu', "the boy", (normally written in Arabic as one word) as '*:alo waladu', "*the boy", thereby segmenting the word into two separate morphemes at sentence level, in order to conform to CHOMSKY's, 1957: 26, Noun Phrase rule: [NP [T N]]. Such analyses are often taken for granted and look innocent enough. In fact, they are ad hoc (since they make no explicit statements about the method of segmentation used nor about how to integrate their morphological and syntactic analyses and which applies first and where and how); but they also constitute a prima facie violation of the criteria of isolability (a bound morpheme cannot occur as a separate item) and insertion (cf., e.g., ALLERTON, 1979: 211). Thus, we could not insert an Adjective: compare the ungrammatical Arabic, '*:alo kabiyru waladu', with the acceptable English, "the big boy". Moreover, since SM is not an explicitly-formulated approach to morphology (rather, it is an expedient method for the segmentation of words in order to perform syntactic analysis), it does not provide any constraints which will stop the generation of examples like '*:alo waladu+', "*the a boy", and '*:alo waladuhu', "*the his boy" (which violate DEFINITENESS conditions in Arabic (cf. Ch. 6, § II.1.6)) and examples like '*:alo rajulu', "*the man", and '*waladu hi', "*his son" (which respectively violate *assimilation* and *compatibility* conditions in Arabic (cf. Ch. 6, § II.1.6.1 and § II.1.6.2)).

Moreover, the inadequate view of grammar, which was expressed in CHOMSKY, 1957: 107 (as cited above), influenced approaches to Arabic MA, such as ERICKSON, 1965, who applies TM to a diachronic study of morphophonemic aspects of English and Arabic verbal morphology. Consequently, his analysis is merely a set of rules, including PS, transformational, and morphophonemic rules (pp. 100–115). Further, as ERICKSON himself puts it, ibid. vi, "the sets of rules are designed to generate the respective [English and Arabic] systems of verbal morphology within a minimal syntactic [my emphasis] framework".

As for CHOMSKY's 1965 view of grammar, it is espoused by LEWKOWICZ, 1967: 62, 142, et passim, by KILLEAN, 1966: 1–2, et passim, and notably by BAKALLA, 1979 who proposes, ibid. xi-xii, an analysis, which is mainly phonological but also morphological, for the derivation of the Meccan Arabic Verb in terms of CHOMSKY's 1965 definition of grammar. However, BAKALLA, ibid. xvi, attempts to remedy the defects of this definition, by adding a morphological component with at least three different sets of rules:

derivational rules which "produce mainly the consonantal bases",

inflectional rules which "develop such bases into fully-fledged verb forms", and

redundancy, or *readjustment* rules, "which state the redundant features of the segments".

However, while attempting to define his morphological rules, BAKALLA only succeeds in landing in a state of incredible confusion. He falls into several serious contradictions and never really succeeds in situating his morphological rules within the 1965 generative grammar framework: the derivable Verb stems are to be, simultaneously, stored in the lexicon, generated by the morphology and the phonology, and treated by the syntax.

Thus, in one place (ibid. xii), he defines a *lexicon* which is part of the Base, and which contains, inter alia, roots and "all possible stems that are derivable from each root". Yet, in a second place (ibid. 657–658), he describes *rule schemata*, which are characterized as "highly abstract in nature" and which stipulate, according to BAKALLA, ibid. 652–653, "that all the Imperf[ective] verb stems take prefixes as well as suffixes at the underlying level of representation". (The morphological rules are now part of deep structure.) However, in a third place (ibid. xvi), he states that the inflectional rules, which, it will be recalled, are part of the morphological component, are to "generate the verb stems and their various items". Then, he defines, loc. cit., a *phonological component* which has "rules that convert the output of the above component into final surface form". He then concludes, ibid. 666, that "the morphological and phonological components form part of the surface structure of the language". (The morphological rules are now, at once, in the lexicon, in deep structure, but also part of surface structure). In yet a different place (ibid. 647), BAKALLA finally stipulates that, in order for the Verb stem to be generated, it has to "undergo further morphological operations, depending on its syntactic contexts". (The morphological rules are now operating on the syntactic level.)

This extreme confusion can only mean that, in the spirit of CHOMSKY, 1965, BAKALLA implicitly views morphological rules as generative transformations. His proposal thus turns out to be a mere variation in terminology: to rename the *readjustment rules*, which link the syntactic component with the phonological component, as *morphological rules*.

I.3.2 The Lexicalist Approach

The recognition of the defects of TM led to the development of the lexicalist approach to morphology, which we will dub *Lexical Morphology* (henceforth LM). LM is a generative school of thought which advocates the treatment of morphology as "entirely within the lexicon" (SCALISE, 1984: 17). It is mainly due to CHOMSKY, 1972b. He suggested, ibid. 17, what he called the *Lexicalist Hypothesis* in order to deal with some aspects of *nominalization*, namely the creation of "derived nominals". In particular, CHOMSKY, loc. cit., argues that "we might extend the base rules to accommodate the derived nominal directly [...], thus simplifying the transformational component". The Lexicalist Hypothesis thus absolved syntax, and in particular the transformational component, of the responsibility for morphology. Instead, the lexicon was now to be filled, according to CHOMSKY, 1981: 5, with "the abstract morpho-phonological

structure of each lexical item and its syntactic features, including its categorial features and its contextual features". This position was echoed, with very little change, in CHOMSKY, 1982: 4–5, adopted by JACKENDOFF, 1977: 2–3, and RADFORD, 1981: 141, and 1988: 337, and is also defended by JENSEN, 1990: 133.

ARONOFF, 1976: xii, describes a theory of morphology which essentially adopts CHOMSKY's 1972b Lexicalist Hypothesis but takes it a step further. He proposes, ibid. 6, that "derivational morphology is isolated and removed from the syntax; it is instead dealt with in an expanded lexicon". The rules, which derive words from words, are generative transformations called *Word Formation Rules* (WFRs) which are concerned with derivation and compounding, and perform phonological, syntactic, and semantic operations (cf. ARONOFF, ibid. 22 and 46). However, WFRs turn out to be too powerful: ARONOFF, ibid. 87, states that "in certain instances the output of a WFR must undergo adjustment before the rules of the phonology may apply". This leads him to propose a number of well-formedness restrictions which constrain WFRs and which are essentially phonological conditions concerned with what ARONOFF, 1976, calls *truncation* (which deletes certain morphemes) and *allomorphy* (which adjusts the shape of certain morphemes). However, SCALISE, 1984: 197, recognizes that "a number of problems remain to be resolved" in the area of constraining WFRs.

Having moved morphology to the lexicon, TG had to find a model for MA within the lexicon. The obvious solution was to merely transfer, or generalize, to the grammar of words, concepts and mechanisms developed by generative theories for syntax. It was then possible to kill two birds with one stone: preserve the universality of syntax, and claim a new universality for morphology, the latter corroborating the former and vice versa. Thus, DI SCIULLO and WILLIAMS, 1987: 22–23, argue that "words have heads", that "suffixes have argument structures [i.e., subcategorization frames]", and that "word formation rules are phrase structure rules". Similarly, SELKIRK, 1982: 1, proposes to examine "the syntax of words" and claims, ibid. 2, "that word structure has the same general formal properties as syntactic structure and, moreover, that it is generated by the same sort of rule system". Hence, she argues, loc. cit., that "just as it is the appropriate formal device for generating syntactic structures, a context-free grammar is appropriate for characterizing the notion 'possible word structure of L'." and that "certain fundamental notions of the so-called $\bar{X}$-theory of phrase structure (i.e., S[entence]-Structure) can be profitably extended to the theory of W[ord]-Structure". We cannot concur with this position, since SELKIRK admits, ibid. 2–3, that her arguments "are founded in large part on the W-Structure of English and related languages", and that "Semitic derived verb forms thus have no immediate constituent structure". In other words, CFGs describe a configurational system (for the syntax of both sentences and words). Since Arabic is non-configurational in syntax (cf. CHOMSKY, 1981: 127–128, SAAD, 1982: 11, ALFEHRI, 1985: Vol. I, 105, fn. 4, all of whom argue (as cited in Ch. 12, § I.4.2) that there are no hierarchical structures in Arabic,

such as VPs) and in morphology (we will show that words in Arabic have a flat rather than a hierarchical structure), we claim that hierarchical descriptions and CFGs are inadequate as an account of Arabic word structure.

I.3.3 The Interpretive Approach

The interpretive approach to morphology, which we will dub *Interpretive Morphology* (henceforth IM) is a generative school of thought due to ANDERSON, 1982, and which advocates the position that morphology is interpreted in the syntax. ANDERSON, ibid. 573 et passim, and 1986: 2 et passim, attacks CHOMSKY's Lexicalist Hypothesis and invites us to construe morphology as a system of rules rather than as a grammatical component per se. After asking: "where's morphology?", ANDERSON, 1982: 611, tries to defend, and elevate to theoretical status, the original position of generative grammar "that there is no completely isolated, uniquely 'morphological' component of the grammars of natural languages". However, his treatment fragments the operation of morphology between the lexicon, the phonology, and the syntax. In his own words (pp. 610–611), ANDERSON asserts:

> morphology is to be found in more than one place. Some of it is in the lexicon, where we find the principles for composing complex stems out of other stems by derivational processes. Another portion is to be found in the syntax, where the principles for constructing morphosyntactic representations are localized. Finally, the rules of inflection, which derive a morphologically complete surface word form [...], are to be found in the "phonological" interpretive component.

However, ANDERSON himself concedes, ibid. 609, that "the alternations among verb patterns characteristic of Semitic languages" as well as other kinds of "stem-internal changes" constitute a prima facie challenge to his claim. Furthermore, ANDERSON's suggestion, ibid. 592–593, that irregular inflections simply be listed in the lexicon (so that inflection is no longer in phonology alone) may prove to be costly and uneconomic in a language, such as Arabic, where irregularities abound, and also poses a problem for his theory: ensuring that (irregular) material which is already treated in the lexicon, is not handled by regular inflection in the phonology. Moreover, ANDERSON's account is observationally inadequate on many counts.

First, ANDERSON's theory, 1982: 609, is built on the distinction of derivational affixes, assigned to the lexicon, and inflectional affixes, assigned to the syntax. However, as noted by JENSEN and STONG-JENSEN, 1984: 479 and 495, ANDERSON's distinction entails stipulating "which categories are inflectional and which categories are derivational for each individual language" and fails to "allow for the case where inflected forms are used as the basis for derivation". ANDERSON's distinction also disregards languages such as Arabic,

where derivation is not only affixational but also *root-and-pattern based* (cf. our discussion in § II.3). In addition, certain Arabic antefixes and suffixes, (e.g., ':a', (question operator), 'wa', "and", and 'hu', "his", in ':awakita| buhu ...?', "and is his book ...?") are, intuitively, neither inflectional nor derivational morphemes.

Second, ANDERSON's central argument, which is based on some examples relating to agreement and the assignment of CASE (mainly in Breton), is that syntactic rules must "have access to the properties of words" (p. 575). However, as JENSEN and STONG-JENSEN, 1984: 474, point out

> syntactic rules can refer to morphological features but need never be sensitive to word-internal structure. For example, agreement rules must refer to features of case, number, gender, etc., but not to the specific suffixes or prefixes that carry these features.

JENSEN, 1990: 115, and JENSEN and STONG-JENSEN, 1984: 477, also correctly point out that ANDERSON's model is inherently contradictory for languages such as Latin, where information about declension and conjugation class determines inflectional endings, (i.e., at the phonological level): if such information is inflectional then it is irrelevant to the syntax, if it is derivational then the phonology has to refer to both inflection and derivation, which is against ANDERSON's grammatical "division of labour".

I.4 RECENT APPROACHES TO MA

The theory of autosegmental phonology has recently been suggested as an alternative model for MA. We will dub this approach, which is mainly due to McCARTHY, 1981 and 1982, *Autosegmental Morphology*. McCARTHY, 1981: 373 and 382, and 1982: 191–192, attempts to explain Arabic Verb derivation, and in particular nonconcatenative aspects of Verb morphology, such as *reduplication* and *infixation*, within the framework of autosegmental phonology. For this analysis, he posits, 1981: 389, 392–393, et passim, and 1982: 192, three simultaneous *levels* (also called *tiers, templates,* and *melodies*): one tier contains the consonants, or root, of the Verb; a second tier contains the vowels of the Verb (i.e., the pattern); and a third dynamic prosodic template contains material such as consonant-affix characteristics of the morphemes. The whole structure is a complex representation wherein the three tiers interact through *association* rules, an interaction whose output is the derivation of Arabic Verbs (1981: 387–389). In order to control the interaction of the three tiers, McCARTHY, 1981: 383, and 1982: 194–195, is forced to formulate a number of rules which relate them. McCARTHY, 1981: 403, is also forced to impose extra rules to constrain his MA which include a specification of the appropriate prosodic templates and affixes for each pattern (p. 392), "a list of triconsonantal roots" (p. 393), as

well as a lexicon for irregular roots (p. 403). Such additions are not without artificiality and unnecessary complication. Hence, HUDSON, 1986: 97, criticizes his analysis for being "rather inexplicit, arbitrary and unnecessarily complex". He argues, ibid. 99 and 119, that McCARTHY actually requires more redundancy rules than he states in order to exclude invalid analyses. Similarly, FINCH, 1986: 180 and 186–187, criticizes McCARTHY for requiring considerably more than the three tiers, which he postulates in his analysis, to be able to deal with Arabic Verbs; as well as for needing "complex and costly" extra rules, in order to handle association between the tiers properly.

Moreover, McCARTHY's rules of affixation assume precise positions for each morphological element: he states explicitly, ibid. 393, that "the rules of association must indicate where the affixes are to be fixed on the prosodic template". Thus, he treats morphological structure as if it were constant: he disregards semivocalic changes in radical elements (whereas the deletion of semivowels, for example, does change element positions in a morphological string), thereby ignoring Defective roots. Hence, HUDSON, 1986: 105, criticizes him for requiring "indexing of the consonants" in order to preserve the invariant ordering of root elements, especially after movement rules. A similar criticism is levelled at him by FINCH, 1986: 191, also on the grounds that "ambiguity arises as soon as the number of *C*'s in the template exceeds the number of consonants of the root". Furthermore, McCARTHY does not account for Arabic antefixes and enclitic suffixes, which do affect morpheme positions and would complicate his analysis. Moreover, his analysis, 1981: 405, entails that morphological rules should not affect "more than one segment at a time", an assumption which does not take cognizance of intermorphemic dependencies, such as vowel harmony in Arabic (cf. Ch. 4, § II.4.3). McCARTHY's analysis has the effect of reducing the root and pattern morphology of Arabic to affixational analysis on the phonological level. It has also been criticized by HUDSON, 1986: 88, for simply reintroducing transformational rules in the guise of "autosegmental notation". HUDSON argues, ibid. 103, that "the translation of deletion, movement and feature-changing rules into autosegmental formalisms does not change their transformational nature".

However, McCARTHY's 1981 analysis has influenced a number of studies, notably FINCH, 1986, HAMMOND, 1988, YIP, 1988, and ABD-RABBO, 1990. Thus, although FINCH criticizes McCARTHY, he still adopts, ibid. 192–195, the autosegmental framework and consequently suffers from the same defects as McCARTHY: the need for indexing rules and for controlling association. Both HAMMOND, 1988, and YIP, 1988, can also be criticized for the same shortcomings as McCARTHY's (YIP, in particular, 1988: 556, simply states: "I accept without argument what he [McCARTHY] showed").

I.5 A CRITICAL SYNTHESIS OF PREVIOUS APPROACHES TO MA

As far as Arabic is concerned, most modern morphological studies focus on philological analyses of the processes of word-formation in Arabic, such as derivation, compounding, and borrowing (cf., e.g., ANIYS, 1951: 8–131, and STETKEVYCH, 1970: 7–65). Further, many modern attempts remain partial analyses and lack completeness. Thus, TRAVIS, 1978: 48, limits her study mainly to inflectional processes of Arabic word-formation, while MAHADIN, 1982: 12, and ELTIKAINA, 1982: 1, are examples of partial analyses of Arabic morphology and which propose to investigate only the derivational aspects of Arabic Verb formation. Moreover, modern studies have retained, for the most part, the traditional view of morphology. Thus, ELTIKAINA, 1982: 73, fn. 1, argues that Verb-bases are modified by affixation and phonological processes to derive other Verb forms. Similarly, MAHADIN, 1982: 12, 16, et passim, argues for the morphophonemic derivation of Triliteral Arabic Verbs from basic forms by affixation and "subject to certain phonological rules".

As for generative theories, their view of morphology (like that of traditional Arab grammarians) remains today still fragmented and confused, besides remaining rather dismissive of Semitic morphology. In particular, with ANDERSON, 1982, TG scattered the regularities of morphology between syntax and phonology, while its irregularities were turned over to the lexicon, in the guise of lists of unrelated lexemes. This fragmented view of morphology may be due to the phonological background of CHOMSKY's formative career and to the fact that CHOMSKY's starting-point was the English language, with its limited morphological system, in formulating linguistic generalizations. Thus, ARONOFF himself, 1976: 4, recognizes that "post-Syntactic Structures linguistics saw phonology and syntax everywhere, with the result that morphology was lost somewhere in between".

Furthermore, another problem in generative work, as noted by MATTHEWS, 1974: 232, is "to ensure that the right rule always applies at the right time". He explains, ibid. 232–233, that it would be undesirable, for generative grammars, if rules were to apply at the wrong time, or if they were to apply more than once. These are the notorious problems of rule ordering and rule constraining. In this respect, ANDERSON's thesis, for instance, faces serious difficulties: organizing the rules, controlling unpredictable interactions among them, ordering them, and ensuring that they do not apply at the wrong time or apply more or less times than they should (cf. ANDERSON, 1982: 608 and 610, and 1986: 3, 6, 14, et passim). Similarly, McCARTHY, 1981, HAMMOND, 1988, and YIP, 1988, all devote most of their papers to considerations in the organization of their model: namely the difficulties of association between the various tiers in templatic morphology, the direction of this association, and the ordering and control of rules (cf. YIP, ibid. 551 et passim, HAMMOND, ibid. 261, 264–265, and 267, fn. 18,

and McCARTHY, 1981: 401 et passim, and 1982: 216 et passim). BAKALLA, 1979: vii, xvii, xvi, and 647, also admits that his analysis suffers from several problems centring around the ordering of rules which, he confesses, ibid. xvii, is sometimes "highly arbitrary and is, therefore subject to revision".

II AN ALTERNATIVE ANALYSIS: A LINGUISTIC MODEL OF ARABIC MORPHOLOGY

Before we start the outline of our own linguistic framework for the MA of Arabic, we have to deal with certain issues which we faced in deciding on that framework. The issues relate to inflectional and derivational levels in Arabic Morphology as well as the choice of a derivational source for Arabic derived words.

II.1 INFLECTIONAL AND DERIVATIONAL LEVELS IN ARABIC MORPHOLOGY

Morphology is usually divided into two major subfields: *inflection* and *word-formation*. Word-formation is in turn divided into two smaller subfields: *derivation* and *compounding* (cf. MATTHEWS, 1974: 38). In traditional Arabic grammar, the term '*:iʕora|b*',"inflection", refers to "the variation in the terminations of words, be it on the surface level or on the abstract level, according to the change of their syntactic functions in the sentence" (YAʕQUWB, 1986: 85). YAʕQUWB, ibid. 85–92, includes in such terminations vowel marks and suffixes which indicate CASE, MOOD, and NUMBER. The term '*:isvotiqa|q*', "derivation", however, refers to "the formation of one word from another, provided that both words are homogeneous in meaning and constituent structure [root elements] but different in surface form" (cf. ALJURJAANIY, 1978: 27). In modern Western linguistics, *inflection* is defined by MATTHEWS, 1974: 74, as "the entire process, or [...] any part of the process, by which a word-form is derived"; while *derivation*, is defined, by ALLERTON, 1979: 215, as "the process by which derivational affixes are added to stems (including simple roots) to form a derived word".

ALLERTON, 1979: 214, draws this distinction (also defended by ANDERSON, 1982: 573 et passim), between the inflectional and derivational levels: inflectional affixes "play a part in expressing syntactic relations between words, such as concord and government, while derivational affixes do not". Further, for ALLERTON, loc. cit., derivational affixes "determine the major syntactic class of the word they form", while inflectional affixes "leave the major class unchanged". Another position adopted by ANDERSON, 1982: 609, and which ALLERTON argues, loc. cit., is that "derivational affixes tend to occur nearer to the root, [and] inflectional ones nearer to the outside of a word". However, ANDERSON himself, 1982: 585–586, raises some objections to the distinction of inflection and derivation: it is generally believed that inflection

is more productive, yet "certain derivational processes are apparently completely productive" (p. 585); it is generally believed that derivation is class-changing, yet some derivational processes "do not have the effect of changing word-class membership" (p. 586); and finally inflection is sometimes defined as a list of categories (such as CASE, NUMBER, and GENDER), yet "the same category may be derivational in one language and inflectional in another" (p. 586).

As far as Arabic morphology is concerned, it is important to underline a number of points about the distinction of inflection and derivation: It is useful to note that while both derivation and inflection are mainly affixational in English, they are not so in Arabic: Derivation in Arabic is mainly root-and-pattern based (cf. our discussion in § II.2 below) but affixation can be both inflectional and derivational. We shall use the term *inflection* (in a similar sense to that defined by YAEQUWB above) to refer to the addition, to a derived word, of affixes (including vowel marks, prefixes, and suffixes) which indicate not only CASE, MOOD, and NUMBER, but also PERSON, TENSE, DEFINITENESS, and PLURAL CLASS. We shall use the term *derivation*, especially in our computational analysis, to refer to the process whereby an Arabic word is created using a derivational source (the root) and a derivational pattern. We will also use *derivation* to describe the process of formation of GENDER categories. In this respect, it is also useful to remember our earlier remark that certain Arabic affixes (e.g., ':a', (question operator), 'wa', "and", and 'hu', "his", in ':awakita|buhu ...', "and is his book ...?") are, intuitively, neither inflectional nor derivational morphemes. The above definitions and distinctions are sufficient for the purposes of the morphological and computational analyses of Arabic carried out in this thesis. However, the beginning of this section shows that the distinction of inflectional and derivational levels has several other theoretical implications. Since our analysis does not hinge upon such theoretical implications, we shall leave this question open for further investigation.

II.2 A DERIVATIONAL SOURCE FOR DERIVED ARABIC WORDS?

Much of the traditional and modern discussions of Arabic morphology has focused on the issue of determining a basic source for the derivation of Arabic words. Such discussions seek an answer to the question: which is the Arabic derivational basic form: is it the *root*, the *stem*, or the *word*? Traditional Arab grammarians are usually divided between two schools. The ALKUWFA school (also followed by McCARTHY, 1981: 402 et passim, ELTIKAINA, 1982: 3, and FINCH, 1986: 181 and 185) consider the perfect Form I (i.e., the *Basic Triliteral* form of the) Verb to be the derivational source for other Arabic words. The ALKUWFA argue that the meaning of the Infinitive reinforces that of the Verb; that the Verb is more important than the Infinitive: syntactically, the Verb governs the Infinitive (as a Subject or Agent) and phonetically, the Infinitive has to follow the pattern of phonetic changes of the Verb; and that some Verbs

have more than one Infinitive while others have none (see also DHAYF, 1979: 196 and 239).

However, the ALBASRA school consider the Infinitive to be the derivational source for other words. For instance, SIYBAWAYHI, 1970: Vol. I, 1, states that "Verbs are instances of forms which are derived from the words that are *Nouns of Events* [i.e., Infinitives]"; while IBNU JINNIY, c.1913: 63 and 79, refers to perfect Verbs as "coming from" Infinitives. The ALBASRA argue that the meaning of the Infinitive is always in the Verb but not vice versa; that Verbs denote an action and a specific time while Infinitives denote a pure action in absolute time; and that not every Infinitive has a corresponding Verb (see also DHAYF, 1979: 137 and 196).

A variety of other derivational sources have been suggested by Western and other linguists. Thus, ARONOFF, 1976: 20, proposes a word-based hypothesis, wherein "a new word is formed by applying a regular rule to a single already existing word". Similarly, McCARTHY, 1981: 375, assumes that "the lexicon is composed of words rather than morphemes". Following this suggestion, ALI, 1988: 277, maintains that "the word is the source of derivation" in Arabic, simply on the grounds that this approach is advocated by ARONOFF, that word-based derivation is less formal than other alternatives, and that "syntax is word-based not root-based". However, he does not demonstrate that any of his claims obtain for Arabic. Similarly, MAHADIN, 1982: 14–15, 148, et passim, argues that the basic form, "from which other forms can be derived, is [the stem of] the third person singular masculine of the imperfect [Verb]". His analysis is based on ARONOFF's 1976 suggestion that derivation is word-based and on phonological arguments, the essence of which is that it is simpler and more economical to derive the imperfect from the perfect than vice versa (pp. 171–173), because there are more (phonological) changes in the perfect than in the imperfect (p. 208) and because the second vowel of the imperfect is (phonologically) unpredictable, were it to be derived from the perfect (p. 301). However, the simplicity that MAHADIN claims is not attested by the number of rules he requires for his analysis. He concedes, ibid. 190, that "to derive the perfect, the imperfect prefix is deleted, the ablaut rule is applied and the appropriate suffix is provided". This is in addition to the need for rule ordering (cf. pp. 192–194) and the need to formulate the precise conditions for consonantal and vocalic changes (pp. 163–171) and for irregularities, which he simply lists in the lexicon (p. 314).

However, TRAVIS, 1978: 90, argues against ARONOFF's 1976 proposal that derivation is word-based and describes it as "untenable" for Arabic. She argues instead, ibid. 12 and 54–55, that "all nouns and verb forms in MSA, in fact, are derivationally related to a lexical root" on the grounds that the root corresponds to our intuitions about Arabic words and that it has "psychological reality". COHEN, 1970: 32, had already argued that, in Arabic, "the derived form is morphologically based, not upon an actual word or upon a given thematic base which is derived from this word, but upon the abstraction which is the root". However, neither

TRAVIS, 1978, nor COHEN, 1970, offer any formal specifications for identifying roots in a fully inflected Arabic word, especially if that word contains antefixes and enclitic suffixes.

Therefore, we object to the derivational sources suggested above for Arabic word-formation on several grounds:

1) Form I of the perfect Verb has to be rejected as a source since there are roots which are not realized in Form I (cf. '*galaqa', ':agolaqa', "to close"; '*qalaʕa', ':aqolaʕa', "to take off", and '*safara', 'sa|fara', "to travel").

2) Infinitives have to be rejected as sources since they have numerous and immensely variable and non-uniform patterns (cf., e.g., WRIGHT, 1896: Vol. I, 110–112) which would require a huge number of transformational rules in order to derive Verb forms from them. This defeats the purpose of simple derivational sources. Furthermore, not every Verb has an Infinitive (cf. 'layosa', '*la:os', "not being"; 'niʕoma', '*niʕom', "blessing", and 'bi:osa', '*bi:os', "damning").

3) The inflected third-PERSON singular form of the imperfect Verb has to be rejected as a source since there are perfect Verbs which have no imperfect counterparts (e.g., the Verbs in (2) above).

4) Words have to be rejected as derivational sources since, in Arabic, this proposal begs the questions: what is the eventual source of any word, what are the adequate criteria for establishing a proper source, and how to construct the source itself. Supposing we were to take the masculine singular Nominal as a source for Arabic Nominals, we still have to decide how to derive the singular itself. We also have to find a source for feminine Nominals that have no masculine (e.g., 'h-adiyqat', '*h-adiyq', "garden") and for dual and plural Nominals that have no singular (e.g., ':it-ona|ni', '*:it-on', "two"; 'nisa|:', '*niso:at', "women"). Furthermore, even if we solved these problems, we still have to solve the problem of deriving the other CATEGORIES, such as Verbs, from the singular Nominal. However, there are Verbs that have no Nominal (e.g., 'layosa', '*layos', "not (to be)") and there are Nominals that have no Verb (e.g., 'sitt', '*yasittu', "six").

Since all derived words in Arabic have one or more derived forms but not all of the possible derivational CATEGORIES and since some derived targets are predictable from the source and others are not, the word-based proposal is too complicated and uneconomic to implement in Arabic. However, a useful generalization that has not been exploited so far is that every Arabic derived target has a root and a pattern. Therefore, we opt for the solution proposed by COHEN, 1970, and TRAVIS, 1978, and that is to use the root as the derivational source for Arabic words. However, we differ from them both by using the pattern, in conjunction with the root, in order to constrain lexically the realization of Arabic derived words.

II.3 A FORMAL SPECIFICATION OF THE GRAMMAR TERMS

In order to account for all the possible and grammatical words of Arabic, it is possible, theoretically, to just list them, as in a WP model, but this approach is trivial, uninteresting, and uneconomic. It is better to define a more general and economic device, such as a formal morphological *grammar*, which will adequately describe Arabic morphology by generating "all and only" the words of Arabic.

II.3.1 Concerning the Definition of Word

Concerning the definition and identification of *word* CRYSTAL, 1980: 383, notes that "there are several difficulties in arriving at a consistent use of the term". Here, we shall use this term simply to refer to an independent *graphological* unit of the written Arabic language which is set off by spaces at either side. (We define *graphological* after CRYSTAL, 1980: 169, as that level of analysis concerned with the study of the writing system and "the minimal CONTRASTIVE UNITS of visual language—defined as GRAPHEMES".) Each Arabic word contains one or more bound *morphemes* affixed to a main segment called a *centre* and each centre is composed of a discontinuous morpheme called a *root* intercalated with another discontinuous morpheme called a *pattern* (we define roots and patterns below), in which case the word will be a Verb or a Nominal. Otherwise, the word is an independent free morpheme, which may be a Particle or a function word that may also have one or more affixed bound morphemes. We use the term *morpheme* after CRYSTAL, ibid. 231, as "the smallest functioning unit in the composition of words".

The morphemes of Arabic are stored in a *lexicon*, or *dictionary*, of lexical elements which have one or more allomorphic variants. We mean by *lexicon* simply an inventory wherein such morphemes are stored. We use the term *lexical*, in this thesis, in the sense of the Arabic term '*lafod/iyy*', meaning "specified, realized, or actualized as a surface form"; and we use the expression *lexical specification* to refer to the features required in a given lexical item by another one immediately preceding it. The lexical entry for each morpheme consists of:

1) A *graphemic* representation including a surface representation for the radical elements, (or *root*) and a semi-abstract representation for the pattern, for Verbs and Nominals; and a surface representation for Particles and function words; and

2) Linguistic *properties*, or *features* covering up to 19 possible domains, (also defined in Degachi, 1989: 85–86), including morphosyntactic features, e.g., CATEGORY, NUMBER, GENDER, PERSON, TENSE, VOICE, TRANSITIVITY, FLEXION, DEFINITENESS, CASE MARK, MOOD MARK, CASE GOVERNMENT, MOOD GOVERNMENT, PLU-

RAL CLASS, TYPE (of root), and MODE; and morphosemantic features, e.g., REFERENCE, HUMANITY, and NUMERICAL VALUE.

Furthermore, each morpheme has internal elements, or *graphemes*; each bound morpheme and each function word have finite predictable length, internal ordering, and predetermined positions. Each Arabic word is formed by concatenation of the various morphemes to each other. This concatenation is not totally free but governed by precedence and other grammatical conditions (explained in this thesis). Any word conforming to the grammar of Arabic morphology is a morphologically *grammatical*, or *well-formed*, string of Arabic. Any string violating one or more rules of the formal grammar is an *ungrammatical*, or *ill-formed*, word. Moreover, words are vocalized (i.e., represented with complete diacritics ('svakol')) and transcribed into the Roman alphabet using the transcription symbols defined at the beginning of this thesis. Note here that ALBAKKUWSH, 1973: 51, fn. 8, argues for the full integration of the vowel marks into the writing of the word and their representation "following the consonants, not under nor over them as is the practice in the Arabic writing system". In this thesis, we shall transliterate words, as if they were completely vocalized and represent all diacritics including the 'svaddat', "gemination", 'h-arakalt', "vowels", and 'sukuwn', "quiescence". Note that even if vowels are dropped in the original text, vocalization really does constitute a cheap form of pre-editing for the purposes of computational work, since it can be performed by any competent Arabic speaker, unlike other forms of computational pre-editing which require specialist training and are generally regarded as inefficient (see CARBONELL and TOMITA, 1987: 73).

II.3.2 Concerning the Definition of an M-Grammar

We use the term *grammar* in the sense defined by MATTHEWS, 1974: 217, as "a set of formal rules", and the term *generate* in the sense given by SELKIRK, 1982: 67, of "assigning a structural description to words". The grammar assigns the proper structural description in the form of descriptors called *Morphology Trees*, or *M-Markers*, to the strings that it generates. Here, we stress this sense of generation, namely specifying all possible, or "all and only", morphologically grammatical words of the Arabic language. We therefore do not imply any sense of directional production of a word. Hence our use of terms, such as *feminization, dualization, pluralization, futurization, definitization,* and *genitivization* (cf. Ch. 6), is restricted to a non-directional process of analysis rather than production.

Furthermore, the grammar we propose is a *Context-Sensitive Grammar* (hereinafter CSG), as opposed to a *Context-Free Grammar* (hereinafter CFG). A *CFG* is a grammar with a set of rules which generate (morphological or syntactic) strings without specifying a particular context for them; in other words, the left-hand side of a rule in the CFG can always be replaced with its right-hand side. A *CSG*, however, is a grammar with rules which are constrained in

their generative power by contextual environments in which the strings may appear; in other words, the right-hand side of a rule in the CSG may replace its left-hand side only in the context specified by the CSG. The morphological rewrite rules of our CSG have the general format:

$$W \longrightarrow X\ [+feature1] \quad Y\ [+feature2] \quad (Z\ [+feature3])$$

where W stands for an Arabic word, X and Y are obligatory morphemes, Z is an optional morpheme, and *features* 1, 2, and 3 are morphological features required in the *context* of the rule and which include the features listed in § II.3.1 above; (basing ourselves on a modified definition by CRYSTAL, 1980: 86–87, we understand by *context* simply the specific structural environment—as a collection of morphological adjacent and sequential units—where a given morphological element, whether allomorphic or morphemic, bound or free, may occur. Where we precede this term with a property name or names (as in *NGP context* to refer to "NUMBER/GENDER/PERSON context"), we mean the particular set of property values obtaining in a given structural domain. Our choice of a CSG is due to considerations of generative capacity as explained by LYONS, 1977: App. 1, 157–169. He suggests, loc. cit., that the most powerful type of formal grammar is an unrestricted rewriting system. It turns out that this type as well as a CFG are too powerful for our purposes. We argue (in § II.3.2 below) in favour of a CSG as a model for Arabic word structure.

In addition to defining words and the M-Grammar, the tasks required for the MA of Arabic involve the identification of particular morphological units such as affixes, roots, and patterns, and of their boundaries. Since we cannot afford a listing approach to the representation of morphological data, in which form should they be stored? The question is particularly important because of the dynamic variation in Arabic morphemes such as affixes, roots, and patterns.

II.3.3 The Listing Form of Affixes

In Arabic, there are several types of affixes. Some of these, such as the prefixes of the imperfect Verb, have fixed forms. However, the suffixes of both perfect and imperfect Verbs undergo different kinds of variation ranging from slight to extensive. These allomorphs are selected subject to *Graphotactic Conditions*. Basing ourselves on a definition of *phonotactics* from CRYSTAL,1980: 270, we mean by *graphotactics* the specific arrangements (or "tactic behaviour") of the graphemes of Arabic. Hence, we use the term *graphotactic* to characterize the relationships between graphemes, and the expression *Graphotactic Conditions* to refer to the constraints governing the juxtaposition of given morphemes to the extent of affecting their graphic shape. Here, we are also indebted to the work of GELB, 1970: 73, who introduces the term *morphographemic* (as opposed to morphophonemic) in his description of Akkadian.

The selection of a given allomorph can also depend on the type of Verb in question. For

instance, the allomorph 'awo|' is used only for Verbs that have Defective roots while 'uw|' is used for Verbs that have both Sound and Defective types of roots. Further, each allomorph can undergo more changes upon becoming a *neighbour* to another affix. The changes include deletion and substitution under certain conditions (cf. Ch. 4, § II.4.3). Here we have chosen to list all the allomorphs for each affix in the lexicon and control the selection of the appropriate affixes at processing time. This approach to listing affixes, admittedly, involves an amount of redundancy at the level of the lexicon. However, the alternative method of listing one morphological form for each affix and forming the other allomorphs by modifying that form implies further complication both at the grammatical level and at the level of implementation and would predictably lead to an increase in processing time, while the choice of a slightly redundant lexicon means a small addition to our required size of virtual memory, at a more affordable cost.

II.3.4 The Listing Form of Roots

Formally, we define a *root*, after MATTHEWS, 1974: 40, as "a form which is not only inflectionally unanalysable, but 'derivationally' and compositionally unanalysable also". Furthermore, an Arabic *'jid-or'*, or *':as;ol'*, "root", is a lexical entry which is a discontinuous sequence of consonantal radicals dividing into several different types (described below) and with implicit internal left-to-right order. Taking Verb roots as examples, we find, in Arabic, various kinds of Verb roots ranging from *Basic* (*Sound* and *Defective*) to *Augmented* (cf. Ch. 4). There are clear differences among these. In particular, Sound roots have one fixed form that does not vary in different *Conjugation Forms* (CFs), in different VOICES, TENSES, and MOODS, and with different *Subject Pronouns* (SPs). However, Defective roots have a dynamic form as their semivowels vary from one context to another. (We modify the phonological definitions of *vowels, semivowels,* and *consonants*, by CRYSTAL, 1980: 380, 318, and 82, respectively, to define them graphologically. *Vowels* and *semivowels* are those units which respectively operate at the centre and at the margins of graphemic *syllables*. (In Arabic, *semivowels* (or *weak letters*) are the graphemes: {w, y, |}). *Consonants* are those units which operate at the margins of graphemic syllables and which are not semivowels. We define a *syllable*, graphologically, based on its phonological definition by CRYSTAL, 1980: 343, as a graphemic sequence of the form 'xvx', where x can be a consonant or a semivowel, and v is a vowel.)

The variation of radical semivowels in Defective roots, involves both their deletion and substitution. For instance, the Defective root 'kwn', "to be" which has the following forms: 'kn', in 'kunotu', "I was", where <w> is deleted, 'k|n' in 'ka|na', "he was", where <w> is substituted by <|>, and 'kwn', in ':akuwnu', "I am", where <w> is preserved. The problem in such variation is: do we list the root 'kwn', for example, in all its different forms, i.e., have three lexical entries for one underlying root or do we list it as 'kwn' with one radical

semivowel included and somehow derive the other two forms from it? The first solution loses an economical generalization, while the latter requires complex generative transformations that would complicate both the grammar and any subsequent program that was to implement it. Furthermore, the transformational approach would benefit only the treatment of Defective roots, Sound roots requiring no transformations since they undergo no variation.

We propose a simpler solution and that is to factor out the semivowels in Defective root forms. Thus, we list only the consonantal radicals of the roots, thereby obviating the need for generative transformations. Having dealt with Basic Sound and Defective roots, we now need to tackle the problems posed by Augmented roots. In seeking an optimum listing form for these roots we are faced with the problem of the infixation of extra characters involved in the augmentation of roots for the purposes of intensification or causativization. Unlike the case of the semivowels in Defective roots, we cannot entirely factor out such infixes. This is because the Augmented root has an entirely different pattern distribution. For instance, the Basic root 'ktb', "to write" has the patterns 'cacac', 'acocac' but the Augmented root 'kttb', "to cause to write" has the patterns 'caccac', 'ucaccic'. Thus 'kttb' behaves as if it was a totally different lexical entry from 'ktb'. Furthermore, certain Augmented Defective roots behave like Basic Defective roots in that they are characterized by dynamic variation. Their conjugation is irregular, hence their semivowels undergo deletion. For instance, 'gnny', "to sing" has the form 'gnny' in 'gannayotu', "I sang", where <y> is preserved and the form 'gnn' in 'gannato', "she sang", where <y> is deleted. Other Augmented Defective roots namely Hollow ones, behave like Basic Sound roots in that their conjugation is regular. For example, 'kwwn', "to cause to be" has the patterns 'cawwac', 'ucawwic' for all SPs. Therefore we can distinguish four types of shapes or root forms:

a.i. Basic Sound with regular conjugation and fixed form.

a.ii. Basic Defective with irregular conjugation and dynamic form.

b.i. Augmented Sound and Defective Hollow with regular conjugation and fixed form.

b.ii. Augmented Defective Quasi-Sound and Weak with irregular conjugation and dynamic form.

The solution adopted above for listing roots of type *a.ii.* by factoring out semivowels can also be applied to roots of type *b.ii.* for a uniform and consistent treatment. However, in the cases of both *b.i.* and *b.ii.*, we still have to make use of the Augmented root form, i.e., with the infixes included, because of the difference in pattern distribution for Augmented roots compared to their Basic counterparts. Yet, there is a generalization here that should not be missed which is that the insertion of Augmentative infixes in Basic roots involves also the modification of the TRANSITIVITY values of these roots by PROMOTION to higher values—for causative

roots—or preservation of the same values, for intensive roots (cf. Ch. 4, § I.1.1).

Consequently, we propose a dual solution to the problem of listing forms for Augmented roots as a whole:

1) include the infixes in the listing form as if we were dealing with a different lexical entry for the purposes of data storage, and

2) derive Augmented roots from Basic roots using their division into causative and intensive, for the purposes of TRANSITIVITY ASSIGNment at processing time.

This solution involves a certain amount of unavoidable redundancy but in line with current computational trends, this redundancy affects the lexicon and not the processor. Any further simplification or pruning in root shapes in the lexicon would be at the expense of the processor in that it would require further devices such as generative transformations for the derivation of Augmented roots.

The above discussion provides a neat solution for the listing of root forms, in that it defines a uniform method for the storage of roots in the database. Consequently, an Arabic root can further be characterized as a canonical, or Basic, form which is extractable from a CF irrespective of VOICE, TENSE or MOOD. Such roots can be used together with Verb patterns to constrain lexically the realization of CFs.

II.3.5 The Listing Form of Patterns

The variation described above in root forms remains a problem after factoring out semivowels. Thus, our solution above provides a method for specifying canonical root forms but it does not account for that variation elsewhere in the grammar. For, how do we control the occurrence of particular semivowels in particular CFs or indeed their deletion or substitution by other semivowels? For instance, if we were to take the example of 'kwn' above, how do we generate only 'kn', 'kwn' and 'k|n' but not 'kyn' or 'wkn' or for that matter 'kmn'?

Consequently, we have to compensate for the flexibility that we achieved in specifying roots, by seeking other means of generating "all and only" grammatical CFs. The pattern structure provides a natural framework for tackling these issues. This structure can in fact be analysed on several levels of representation, for instance, as a completely underlying structure or as a completely lexical structure. The introduction of the Arabic morphological '*wazon*', or '*miyza|n*', "pattern" (also known by its Hebrew name '*binyanim*'), is usually attributed (cf. DHAYF, 1979: 35) to ALXALIYL, IBNU Ahmad Alazadiyy (from the ALBASRA school, d. c.786). Specifically, in Arabic, a *V-Pattern*, is traditionally viewed as an abstract arrangement of radicals and vowels in the general form: 'fvɛvl' where {f, ɛ, l, v} are *base* graphological categories, with {f, ɛ, l} representing consonantal and semivocalic radicals and <v> represent-

ing vowels.

Similarly to Arab grammarians, BOHAS, 1975: 355, argues that "all the irregular verbs [i.e., Defective ones] studied here can be derived from an underlying source CVCVC using a small number of [phonological] rules". Similarly, TRAVIS, 1978: 20 and 29, and ELTIKAINA, 1982: 121, fn.*, propose the base forms: 'cvcvc' and 'ccvc' (where <c> represents any consonant or semivowel and <v> represents any vowel) for the perfect and imperfect Arabic Verbs, respectively; McCARTHY, 1981: 386–387, also proposes underlying patterns, which have such forms as 'cvcvc', and which do not distinguish consonants and semivowels; while MAHADIN, 1982: 318, suggests a slightly different solution: 'cacvc' (with one vowel specified) and 'ccvc' for the perfect and imperfect Arabic Verbs, respectively. However, the level of abstraction required for underlying representations, such as the above, as well as those proposed by traditional Arab grammarians, is too deep. In order to get to a lexical pattern realization such as 'katab', "wrote", such abstract representations require an intermediate level, again such as a generative transformational component, in order to derive lexical CFs from underlying representations.

Because of the complications involved in formulating such intermediate levels, Arab linguists have traditionally introduced direct lexical constraints in the specification of patterns, by stating explicitly the precise vowels required for a derivation in the pattern form as in 'faɛ al'. This conventional form, thus, juxtaposes two levels of representation, *base* graphological categorization and *lexical* vocalic structure. Loosely speaking, such forms can be seen as a shallow level of morphological representation, mainly in the sense of being *intermediate* between *deep* and *surface* representation. The realization of this intermediate pattern into completely lexical forms involves the substitution of the base categories for any consonant or semivowel and the exact matching of the vowels specified in the representation. In this sense, the vowels constrain patterns directly, i.e., without any generative transformational rules, whereas the base categories constrain the positions assigned to each radical so that only lexical radical sequences belonging to the language can be generated. Such processes are referred to as *derivations*, hence the expression *derivational pattern* to refer to the intermediate form of representation described above.

However, this intermediate level of representation can be criticized in two respects:

1) it accounts only loosely speaking for the substitution of radicals and does not specify precisely which radicals are to be substituted by consonants and which others by semivowels, and

2) it involves redundancy in specifying derivations as it requires three rules to substitute <f>, < ɛ >, or <l> by a consonant or a semivowel rather than just one rule: since the pattern implicitly specifies radical positions, there is no need for extra explicit position specification. This also happens to be the case for computational analysis in the computer

language LISP where every object has a specified position by virtue of being in a structural sequence (for a description of LISP, cf. Ch. 3, § I).

The above criticisms can be overcome if we find a solution to the problems caused by the conventional representation. Now, we have already seen how traditional pattern representations account for augmentation by sometimes specifying precise radicals for derivation. For instance, the pattern 'fa|εal' requires the insertion of the lexical infix <|> in a lexical pattern in order to augment it to the "meaning" of "participate in", hence 'katab' is "to write" and 'ka|tab' is "to correspond with", 'qatal' is "to kill" and 'qa|tal', is "to fight with", etc. This facility can simply be extended to solve the imprecision described above, by representing only non-terminal consonantal categories and specifying lexical semivowels. Admittedly, this will increase the number of patterns but this affects only Defective patterns, i.e., those that contain semivowels.

In fact, it turns out that this solution also solves the dynamic variation observed above in root forms. Since the semivowels are specified lexically, they can operate, like vowels, as derivational constraints, obviating the need for external constraints and avoiding the imprecision that we find in conventional representations. Since the only non-terminal categories will now be consonants, we can represent these using the symbol <c> for all radicals thus stating only once the substitution rules necessary for pattern matching: that is, substitute <c> for any consonant, not substitute <f>, substitute < ε >, substitute <l> for any consonant. The simplification of the conventional method of pattern representation by removing affixes from it and unifying the consonantal symbols: {f, ε, l} as one symbol: <c> gives us the new format 'cacac-acocac', instead of 'faεala-yafoεalu'. Thus the pattern 'fa|εal' above could simply be represented as 'ca|cac'. We can now give a formal modified definition of *patterns*. A *pattern* is then a shallow structural description consisting of a sequence of consonantal tokens and lexical vowels and semivowels. Together with roots, its function is to constrain the derivation of words. Unlike the conventional representation, the modified pattern sequence excludes in its representation the specification of affixes as these are 1) independent of pattern specification, and 2) already specified in the grammar.

The formal definition of our morphological grammar and the specification of its terms: words, affixes, roots, and patterns, thus provides very specific and precise structural definitions and constraints which can now be used to construct an optimal database and algorithm for the generation and processing of these structures. We will provide an outline of our computational processing in Chapter 2, § III. However, we start by a review of computational parsing in Chapter 2, § I and § II.

==0==<★★★★★>==0==

Chapter 2

THEORETICAL ISSUES IN COMPUTATION

INTRODUCTION

In this Chapter, we deal with theoretical issues in computation. We discuss, in Section I, some relevant aspects of computational parsing and theoretical issues and criteria in the design of a parser. We review the major formalisms which are used in parsing, the techniques for implementing them, and the theoretical frameworks which have most influenced the computational processing of Arabic.

In Section II, we give a critical account of previous computational analyses of Arabic morphology, the formalisms they use, and the theories they are based on, especially insofar as they are relevant to our analysis. We end Section II with a critical synthesis of the previous analyses which explains why they are not satisfactory.

The critical synthesis allows us to propose, in Section III, an alternative computational approach to Arabic morphology. The approach involves the description of search methods in general and of the modified search method which we propose for a morphological analyser of Arabic. It also involves defining the computational tasks of the analyser, its components, and its implementation.

I COMPUTATIONAL PARSING

I.1 THEORETICAL ISSUES IN THE DESIGN OF A PARSER

In discussing *parsers*, we need to deal with some theoretical issues in their design. Before doing so, we need to define what *parsing* is. Following DE ROECK, 1983: 8, we define the term *parsing* by replacing her "sentence" with "word" thus: parsing is a procedure that will,

> not just recognise the word but also discover how it is built. The execution of that procedure is called *parsing* and the thing that executes it is called a *parser*. So, parsers do essentially two things. On the one hand, when presented with a string, they have to recognise it as a word of the language they can parse. [...] On the other hand they have to assign to that word a structure which they have to output.

The recognition of a word involves establishing if it belongs to the language by applying an algorithm to it. In this context, it is also informative to remind ourselves of JOHNSON's 1985 definition of a parser in terms of what kinds of knowledge it needs and makes use of. He distinguishes, ibid. 33–35, four kinds of knowledge, including:

1) "well-formedness conditions" (i.e., a grammar for well-formed input structures),

2) "structure-building rules" (i.e., a procedure for output structures),

3) "knowledge about control" (i.e, how to apply rules and in which order), and

4) "appropriateness criteria" (i.e., *heuristics* to choose the best result if more than one is available). A *heuristic* is defined by DARCY and BOSTON, 1983: 120, simply as "a trial and error method, using rules of thumb, for finding the solution to a problem by evaluating the progress made at steps along the way".

Now, many issues in designing a parsing procedure are linguistic issues, which are concerned with theoretical approaches and analyses. However, there are other technical issues involved in parsing and which are concerned with parsing methods and results.

I.1.1 Practicality vs Theoretical Soundness

Two different trends have recently emerged in computational work and these are the need for practical results in parsing, on the one hand, and the need for a theoretical basis to parsing systems, on the other. Hence, OBERMEIER, 1987: 232, argues that "NLP systems of the future have to be based on more innovative linguistic and conceptual models if they are to become useful and accepted on a large scale"; while WINOGRAD, 1983: 361, points out that systems which are primarily motivated by theoretical considerations "do not emphasize wide coverage or practicality, but they attempt to capture in their design the same kinds of generalizations

that linguists and psycholinguists posit as theories of language structure and language use". However, SAGER, 1979: 159, argues that

> much of the grammatical work that has been done in a formal framework, chiefly in transformational generative grammars, has not led to an integrated grammar of the whole language and does not deal with the issues faced in recognition. Computational linguists have therefore had to write their own grammars, or parts of grammars covering the range of language material their systems are intended to handle.

while GRISHMAN, 1976: 7, remarks that "the last few years have seen most work in language processing devoted to the development of integrated systems, combining syntactic, semantic, pragmatic, and generative components". Similarly, RITCHIE and THOMPSON, 1984: 368, point out that, in efficient parsers, the emphasis is on "having a stable, usable parser that covers a reasonably wide range" and "responds in an acceptably short time". KING, 1981: 44, also stresses the importance of making a computer system produce some result rather than fail completely, explaining that a "system cannot be allowed just to give up"; SLOCUM, 1981b: 1, goes so far as to suggest that "very occasional poor performance may be quite acceptable, [...] provided the overall average performance is superior"; and McDERMOTT, 1982: 148, makes this point informally, but more poignantly, saying that "if 'mechanical translation' had been called 'word-by-word text manipulation', the people doing it might still be getting government money". Thus, the need for adequate linguistic theories has to be balanced against the need for practicality and realism in the design of a parser.

I.1.2 Efficiency vs Uniformity

Uniform systems insist on precision and generality, by attempting to account for all languages (or as many languages as possible). Hence, they may often result in rigidity. Opting for an efficient system which is language-specific and less uniform, can lead to more flexibility. Thus, BARR and FEIGENBAUM, 1981: 257, stress that "in general there is a trade-off between efficiency and uniformity; an algorithm specially designed for only one language can perform more efficiently than one that could uniformly handle any language". In addition, SLOCUM, 1981b: 6, and 1981a: 106, argues emphatically that "primary efficiency gains may come through reduction in the number of grammar rules misapplied more than from decreases in pure parse time alone". Moreover, the efficiency of a system can also be increased by opting for overgeneration at certain points in the program and filtering invalid results later. Thus, HUTCHINS, 1986: 182, notes that "many analysis programs incorporate 'filters' which check for well-formedness of derived structures".

I.1.3 Modularity vs Linearity

The optimization of parsing systems involves choices at the level of their organization, or architecture. The choices deal with the questions of *modularity* (a process by which programs are organized into separate units, or modules, interacting via a main processor) vs *linearity* (a process by which programs are written as a single unit consisting of a sequence of lines, a rigid format which was predominant in earlier programs).

However, as HUTCHINS, 1986: 181, reports, "modularity has become the norm" since the mid-1960s. KING, 1981: 45, points out that the overall modularity of a system means that the "addition of new rules does not perturb the existing set of rules". Similarly, HUTCHINS argues, loc. cit., that "the advantage of dividing procedures into relatively independent subroutines is that parts can be modified and tested"; (DARCY and BOSTON, 1983: 199, define a *procedure* as "a set of rules and steps" which form a subpart of the program and are to be followed in its execution. They define a *subroutine*, ibid. 240, simply as "a sequence of instructions that performs a specific task, usually used more than once in a program").

I.2 A REVIEW OF PARSING TECHNIQUES

There are several methods of proceeding in a parsing system. Such methods vary from one program to another.

I.2.1 Augmented Transition Networks

The *Augmented Transition Network* (hereinafter ATN) is a formalism which is usually attributed (HUTCHINS, 1986: 182) to WOODS, 1970 and 1973, and which has become one of the standard tools of computational linguistics (JOHNSON, 1983: 69). Following JOHNSON, ibid. 61, we define a *simple transition network* as a state diagram in which the *nodes* stand for states and the *arcs* for transitions. *Labels* on the arcs give enabling conditions on the input for a transition to take place. Using this definition and a paraphrased statement from FITCH, 1988: 12, we can start to define an *ATN* as a simple transition network with the additional capabilities to:

1) remember what has already occurred in the parse;

2) manipulate features and conditions on constituents; and

3) add and delete constituents.

WOODS, 1970: 592, describes the extensions which augment ATNs, noting that

> we can add equivalent facilities to the transition network model by adding to each arc of the transition network an arbitrary condition which must be satisfied in order for the arc to be followed, and a set of structure building actions to be executed if the arc is followed. We call this version of the model an *augmented transition network.*

He then describes, ibid. 593, the main tasks of an ATN, stating that "the augmented transition network builds up a partial structural description of the sentence as it proceeds from state to state through the network. The pieces of this partial description are held in registers which can contain any rooted tree or list of rooted trees.". In his description of ATNs, WOODS, ibid. 591 and 597, and 1973: 111, is anxious to point out that the ATN "provides a model which is capable of doing everything that a transformational grammar can do". However, now that the application of generative transformations has virtually been abandoned (cf. our discussion in § I.2.6, below), this has become of questionable merit.

In addition, WINOGRAD, 1983: 248, notes that "ATN grammars are usually written in a style that accepts many questionable constructs in order to have the generality to handle the sentences intended by the designer". This flaw is recognized by WOODS himself who concedes, ibid. 125, that an ATN is too powerful to generate "all and only" grammatical sequences in a language. Moreover, the difficulties in expressing Arabic morphological derivation with roots and patterns in the form of a CFG, represent serious problems for the ATN formalism.

I.2.2 ATN Implementation

There are two types of parsing techniques, or strategies, in which ATNs can be implemented: *deterministic* and *nondeterministic.* A *deterministic* technique is one in which there is always only one way of proceeding at any given point. *Nondeterministic* parsing techniques, in which there may be more than one way of proceeding at a given point, fall into two groups: *top-down* and *bottom-up.*

1) A *top-down* method is a type of goal-directed, or hypothesis-driven, processing which for WINOGRAD, 1983: 90, "is guided by the goals it is trying to achieve", such as the recognition of a word or a sentence. Top-down parsers always start with the starting symbol [of the grammar] ('*S*'), find rules that apply to it and expand it. Such parsers first check whether any of the nodes generated by the grammar is *terminal*, i.e., whether they contain symbols that belong to the vocabulary of the language. If so, those symbols will be checked against the string that is being parsed. If not, the parser further expands the first non-terminal node. This way of proceeding is repeated and after the application of all the rules, the string to be parsed is actually met: no further rules apply and the parse, outputting a structure for the string, succeeds (see DE ROECK, 1983: 11).

However, the top-down method can proceed in a straightforward way only if there is one route to follow at a given point. If not, then the parser can make a mistake in choosing a route to follow, in which case it has to go back to the multiple choice point and start parsing again, which may result in lower level constituents being wastefully analysed again and again, until the correct solution is found. This is called *backtracking*. The disadvantage in backtracking as WINOGRAD, 1983: 63, rightly says, is that the amount of work for the parser

> is exponentially related to the length of the sentence. The number of different paths that could be tried is multiplied by a constant factor for each additional word in the input, since it represents a new choice between a set of possible arcs that is independent of the set of choices made by the previous words in getting to that state.

Similar criticisms of the wastefulness entailed by a top-down method, in general, and by backtracking in particular, are made by HUTCHINS, 1986: 183, JOHNSON, 1983: 60, and TENNANT, 1981: 72, and is also acknowledged by WOODS himself, 1973: 148.

2) The *bottom-up* method is a kind of data-directed processing. This means that it is guided by the availability of specific data such as words. In the words of FITCH, 1988: 7, bottom-up parsing simply uses "the grammar rules backwards" while building back to the category of the starting symbols (such as the word or sentence). WINOGRAD, 1983: 90, expands this further, explaining that

> a bottom-up procedure begins with the words and looks for the rules whose right-hand sides match sequences of adjacent words that can then be combined into a constituent as identified by the left-hand side. It then tries to combine these with each other and the remaining words into larger constituents, and proceeds up the structure tree until it is able to combine constituents covering the entire input into a single structure labeled with the distinguished symbol [*S*].

However, the bottom-up method is not without some major problems. For instance, it cannot make predictions about what to expect next in the parse. PEREIRA, 1985: 317, attempts to defend this method, claiming that "the criticism that bottom-up parsers have no predictive capabilities is unfounded". He goes even further, ibid. 318, when he claims that "the class of languages that can be parsed bottom-up without backtracking is strictly larger than the class of languages parsable top-down without backtracking". However, OBERMEIER, 1987: 228, correctly argues that, since bottom-up parsers are not goal-directed, "they generate numerous spurious parses. And the correctness of the parse can be determined only after all the parses are performed.".

I.2.3 Definite Clause Grammars

Definite Clause Grammars (hereinafter DCGs) are a formalism for writing grammar rules which are mainly due to COLMERAUER, 1978, KOWALSKI, 1980, and PEREIRA and WARREN, 1980. A *DCG* views the rules of a grammar as specific instances of logical forms. Each rule can be translated into a *logical formula* (called a *Definite Clause*) with a number of variables, which refer to sequences of words in the language, and one or more predicates, which describe formal relations between the variables. For each word in the language there are *atomic propositions* representing its lexical categories. (An *atomic formula*, in PEREIRA and SHIEBER, ibid. 9 and 14, is a predication "representing primitive propositions" and expressed in logical notation "using parentheses surrounding the arguments following the predicate".) A grammar, that is translated thus into logical form, can then be parsed using a *theorem prover*. In order to prove a statement from the grammar, a series of *proof steps* must be produced that correspond directly to the parse tree. Thus, PEREIRA and SHIEBER, 1987: 1–2, argue that "if the conditions for a certain class of problems can be formalized within a suitable logic as a set of premises, and if a problem to be solved can be stated as a sentence in the logic, then a solution might be found by constructing a formal proof of the problem statement from the premises".

However, PEREIRA and SHIEBER themselves, ibid. 2, enumerate a number of difficulties with this approach. Namely, they stipulate, loc. cit., that "a proof procedure should not leave proof possibilities unchecked" and that "the proof procedure should be goal directed in that derivations of irrelevant consequences [of a premise] are avoided". They go on to claim, loc. cit., that "the problem then becomes one of finding a good compromise between expressiveness of the logical language and the constraints of sound and efficient computation". Moreover, WINOGRAD, 1983: 350, also concedes that due to the emphasis of DCG on "transparency" and to the fact that, in logic, if a proposition has been proved it stands and cannot be changed, the DCG formalism lacks flexibility, besides presenting technical problems in writing grammars in DCG forms which are immediately perspicuous.

I.2.4 DCG Implementation

DCGs are generally implemented in *PROLOG* (cf. PEREIRA and SHIEBER, 1987: 73); (*PROLOG* is a computer language which is largely declarative: it states what is computed rather than how to compute it, as opposed to a procedural language, such as *PASCAL*, which defines objects in terms of how they are computed (cf. PEREIRA and SHIEBER, ibid. 3, and our discussion, Ch. 3, § I.1)). PEREIRA and SHIEBER, ibid. 35 and 178–179, stress that "Prolog supplies by default a top-down, left-to-right, backtrack parsing algorithm for DCGs".

However, they concede, loc. cit., that "it is well known that top-down parsing algorithms of

this kind will loop on left-recursive rules" and that "although techniques are available to remove left recursion from context-free grammars, these techniques are not readily generalizable to DCGs, and furthermore they can increase grammar size by large factors. As an alternative, we may consider implementing a bottom-up parsing method directly in Prolog.". We have already commented on the disadvantages of both top-down backtracking methods and of bottom-up strategies.

I.2.5 Two-Level Finite State Morphology and its Implementation

Two-Level Finite State Morphology is a formalism for computational MA which is due to KOSKENNIEMI, 1983, and which was specifically designed for the analysis of Finnish, which is an agglutinative language. This formalism is based on a lexicon and posits two levels of representation for words: an abstract, or deep level, which is the lexicon itself, and a surface level. The deep level contains a set of lexical *morphemes* and each *morpheme* is a string of characters (KOSKENNIEMI uses the term *lexical*, in a special sense, to characterize the underlying representation of a morpheme in the lexicon). The surface level contains strings of characters for each word as it is realized in the surface structure. The MA is performed by matching the surface realization with the abstract lexical form. This matching is very intricate and is done on a character-by-character basis, starting with the surface character, and looking for its valid lexical character in the lexicon. The model is so abstract that the deep level, for KOSKENNIEMI, ibid. 23, "may contain some characters which never occur on the surface". In this way, a lexical character may have a zero realization in the surface, and a surface character may have no corresponding abstract lexical character, thus allowing for the vacuous application of rules.

However, this model needs additional rules (used as filters by specifying the exact context for the realization of a given character) in order to block invalid analyses, "to account for all discrepancies between lexical representation and the surface word forms", and to forbid "certain combinations of lexical units" (KOSKENNIEMI, ibid. 30). Moreover, the use of an abstract level adds great complexity to the system exactly like the complications entailed by the deep/surface structure duality in CHOMSKY's theory and the consequent transformational component that maps one onto the other.

In his review of KOSKENNIEMI's model, GAZDAR, 1985: 5, criticizes the model for being "outmoded and inappropriately restrictive". He laments, ibid. 5–6, the facts that KOSKENNIEMI's formalism does not employ "the feature bundles that have been standard in linguistic work on phonology for two-decades"; that it does not employ familiar notation; that it does not have a "straightforward way of talking about tree-structured objects"; and finally that it "can

only describe finite sets of strings that are finite state languages". Obviously, all of these points, apart from the last one, would be major defects in any generative linguistic model. However, if they were to be avoided, this would not, ipso facto, represent inherent theoretical advantages. Nonetheless, an important defect that is admitted by KOSKENNIEMI himself, ibid. 27, is the fact that "some extensions or revisions will be necessary for an adequate description of languages possessing extensive infixation or reduplication". Since infixation and reduplication are prevalent in Arabic morphological structure, it would be spurious to attempt applying the KOSKENNIEMI model to Arabic morphology. In addition, because of the strict, mechanical, and rigid linear configuration of the two abstract and surface levels, it is difficult, in this model, to account for Arabic discontinuous morphemes.

I.2.6 Generative Transformations

Generative transformations (defined in Ch. 1, § I.3.1) and the *combinatorial approach* of Information Science are theoretical frameworks which have particularly imbued the computational processing of Arabic in recent years. The use of generative transformations in modern linguistics is mainly due to CHOMSKY, 1957 and his later works. Because of the influence of TG in modern linguistics, some systems have attempted to implement it on the computer. However, GRISHMAN, 1976: 14–15, rightly points out to three basic problems in implementing a TG. He enumerates, loc. cit.,

1) "assigning to a given sentence a set of parse trees which includes all the surface trees which would be assigned by the transformational grammar";

2) "given a tree not in the base, determining which sequences of transformations might have applied to generate this tree"; and

3) "having decided on a transformation whose result may be the present tree, undoing this transformation".

Hence, in their description of the *Linguistic String Parser*, a system for the syntactic analysis of English scientific texts, FITZPATRICK and SAGER, 1974: 7, recognize that "the same string form has several transformational sources". Similarly, WOODS, 1970: 596, cites PETRICK, 1965, and MITRE, 1964, as two attempts to formulate "practical algorithms for transformational recognition" and comments, loc. cit., that they "resulted in algorithms which were either too time consuming for the analysis of large numbers of sentences or else lacking in formal completeness". Therefore, WOODS, ibid. 600, concludes: "the analysis problem for the transformational grammar is so extremely complicated that no reasonably efficient recognition algorithm for transformational grammar has yet been found". GRISHMAN, 1976: 73, concurs with this conclusion, reporting that implementations of TG imply, "unfortunately, that parsing

time can increase exponentially with the number of applicable transformations. Such a procedure has therefore proved impracticable for all but the smallest grammars and sentences.".

I.2.7 Combinatorics and Information Science

Combinatorial analysis of Arabic morphology has its roots in information theory and its statistical methods which became prominent in linguistic work during the 1950s and 1960s. Examples of such combinatorial analysis can be found in the work of linguists such as HERDAN, 1962a and 1962b, and CZAPKIEWICZ, 1965–1966. Such analyses attempted to apply combinatorial mathematics to word-formation within an information theory framework and then compare possible, or potential, frequencies with actual occurrences in the language. For instance, HERDAN, 1962b: 263–264, based on earlier work by GREENBERG, 1950, and on some equations which are irrelevant to our discussion, calculates that there are 15,600 possible Arabic Triliteral roots, of which only 6,332 are admissible combinations. Of the 6,332 total, he concludes, ibid. 266, and 1962a: 54, that only 3,775, or 60%, are actually used in word-formation. Similarly, CZAPKIEWICZ, 1965–1966, attempts to solve the problem of word lengths in Arabic. He arrives, ibid. 310–313, at a 14-phoneme maximum length for Arabic words but only 6.53-phoneme, or less than 50%, "standard length of Arabic words".

Such statistical studies may be of some value in fields such as text analysis. However, the mathematical generation of all possible combinations produces a vast discrepancy between its results and the results of an "all and only" approach. Note the 9,268 difference between 15,600 and 6,332 in the case of HERDAN's analysis. In addition, there is a degree of uncertainty in the results of such combinatorial analyses. For instance, MRYATI, 1987: 102–103, starting from HERDAN's 15,600 figure (for Arabic Triliteral roots), arrives at 9,097 (as opposed to HERDAN's 6,332) admissible combinations, and concludes, loc. cit., that only 7,198, or 80%, (as opposed to HERDAN's 3,775), are actually found in Arabic dictionaries. Nevertheless, the combinatorial approach to language study was to have a lasting and pervasive influence on the development of early as well as later computational systems for the analysis of Arabic morphology.

II CRITIQUE OF PREVIOUS COMPUTATIONAL ANALYSES OF ARABIC MORPHOLOGY

II.1 EARLY ANALYSES

1) COHEN, 1961: 48–70, describes a MA of Arabic which is intended for Machine Translation. COHEN's description offers a rare insight into Arabic morphology and suggests a number of

useful descriptive constraints as a means of controlling the complexity of Arabic morphological structures. However, in his implementation, which has undergone limited testing, the author does not take full advantage of such constraints to reject invalid analyses. Instead, COHEN, ibid. 64, insists that "we have generally let the algorithm carry on until the end of the analysis", thus producing all possible analyses first, and then putting these through a "correcting circuit", which is essentially another program run which filters out invalid results. This method is very inefficient and has no way of avoiding the repetition of work already done. In addition, COHEN, ibid. 63–64, admits that his method does lead to "absurd results" and an unqualified number of "embarrassing" errors. Such errors were caused mainly by his inability to deal with Arabic Defective forms.

2) The discussion in SATTERTHWAIT, 1962, 1963, 1965, is a typical example of the conception of Arabic morphology in early computational work. This is the conception which we might term, after KOSKENNIEMI, 1983, the "strip-off-the-endings" approach that is characteristic of the MA of English by computer: strip a suffix and look for the remainder of the word which is just listed in a dictionary. The stripping process, as described in SATTERTHWAIT, 1965: 19, allows just one affix per word. With this simplistic view of Arabic morphology, SATTERTHWAIT, 1962, manages to have a running Arabic-English sentence-translation program, with a very limited amount of Arabic MA, actually done. Nevertheless, SATTERTHWAIT, 1963: 62, simply admits that "many of these sentences [that his program produces] can only be termed nonsense sentences".

3) BATHURST, 1971: 185–190, outlines a project started in 1968 in the Middle East Centre in Cambridge University which aimed to alphabetize Arabic words by computer. This project was to be tested by word frequency counts of modern Arabic prose, for which a corpus was built. The approach described by BATHURST, 1971, is theoretically based on combinatorial methods, such as those used by HERDAN for sampling of data, which impose severe restrictions on the type of analysis undertaken. Hence, he complains, ibid. 185, that

> the abundance of prefixed and infixed letters make it impractical to alphabetize Arabic words by taking the letters sequentially. There can be up to 2,300 different combinatory prefixes preceding the first letter of a verb stem and more than 1,600 preceding a substantive stem. The stems themselves are subject to infixion and there are more than 500 patterns for them. Varieties of combinatory suffix are almost as numerous as those of the prefix.

Such combinatorial difficulties forced BATHURST, ibid. 186, to try "to recognize Arabic word-roots (stems without their infixes)", since infixes would interfere with combinatorial possibilities, as well as to the exclusion, as cited above, of any sequential analysis, which would significantly increase combinatorial possibilities.

4) SCHMIDT, 1971: 12, describes a program which aims to generate automatically and from consonantal roots, all Arabic Verb forms. SCHMIDT, ibid. 14, describes an IP model of MA, which is adapted to an approach with a generative flavour, which he calls, loc. cit., a "synthetic, process grammar".

Although SCHMIDT, 1971: 5, claims his "program has been designed to produce all and only the well-formed finite verbs", he concedes, ibid. 59, that "the verb generation program produces all possible verb forms from any root [...] regardless [sic] of whether or not all such forms in fact occur in MSA for the sample root". He then attempts to justify his position, arguing, loc. cit., that, anyway, "standard dictionaries indicate which forms are in current use, although even the non-occurring forms are available to the creative writer [sic] who wishes to begin employing them". If one tried, one could not be more cavalier about a minimum of observational adequacy in linguistic analysis. Having produced all possible forms of the Arabic Verb, SCHMIDT, 1971: 13, then attempts to eliminate "forms which do not occur for a particular class of roots". He is not very precise on the methods used to achieve this, but he does reveal, that he uses certain rules, which are essentially ad hoc, (he calls them "eclectic" (p. 14)). He then admits, ibid. 5, that, in some cases, the ad hoc rules were forced upon him, in the form of "simplifications of sets of traditional rules and revisions prompted by the ordering of the rules", thereby effectively changing the Arabic data to suit his hypothetical ordering.

5) FRAZIER, 1976, claims to have a technique for investigating Arabic morphology. However, his work turns out to be, in his own words, no more than "computer-compatible files [which] are actually "instant Wehr". They are Wehr on punched cards, on magnetic tape, and, [...] in printout." (Vol. I, p. 66). He explains his reference to WEHR, saying, loc. cit., that "I created these [computer] files from the Wehr dictionary to serve as a tool for my own use in my long-range plan to derive all verbal forms from the masdar [Infinitive]". FRAZIER, 1976: Vol. I, 8, 37, and 38, uses three matrices, or computer grids, which are :

- "a grid opposing 31 phonemic segments to 37 phonetic features [distinctive sound classes]",
- "a grid opposing 28 segments (the traditional radicals of Arabic orthography) to themselves", and
- a central file containing mainly "the triliteral [...] roots of the Form I verbs", "the masdar patterns and deviations", and "the tense forms and deviations".

The derivation of the "verbal forms", as described by FRAZIER, ibid. 9, consists in issuing a series of commands to the computer to compute "every possible distribution of each phonetic element and each phonemic segment [as listed in the matrices]". Since, he generates, ibid. 7, all possible forms, he has to filter the invalid combinations by matching the output forms with "2,532 triliteral roots" as entered in WEHR's dictionary.

II.2 MORE RECENT ANALYSES

II.2.1 ATN-Based Analysis of Arabic Morphology

Various techniques have recently been proposed for the computational analysis of Arabic morphology. Most of these (BEN HAMADOU, 1986a, and 1986b, BEN HAMADOU and SAIDI, 1986; DEBILI, 1989; SAIDI, 1985; DEBILI and ZOUARI, 1989; and HLAL, 1984 and 1987) have adopted the ATN formalism for that purpose. BEN HAMADOU, 1986a and 1986b, and BEN HAMADOU and SAIDI, 1986, describe an ATN-inspired "data-compression technique for Arabic dictionaries" which is geared to the "automatic detection and correction of spelling errors in Arabic texts" and which is implemented on a PERKIN-ELMER machine. DEBILI, 1989, and DEBILI and ZOUARI, 1989, describe a morphological generator, which contains an automatically-generated set of dictionaries and a grammar, for the generation of morphological structures to be retrieved from Arabic text. HLAL, 1987, describes a technique for information retrieval (indexing and questioning) of Arabic documentary texts for the purpose of constructing bibliographic files. Unlike BEN HAMADOU, neither DEBILI nor HLAL give any details of the implementation of their work or indeed its testing or performance.

We note that BEN HAMADOU, 1986b: 288, proposes an "affix congruity matrix $C(P, S)$" with two dimensions: 21 prefixes and 29 suffixes, producing 609 possible combinations. Of these, only 226 are valid. The rest have a value $\emptyset$. The infix is obtained separately by a *Morphological Code* (MC) associated with the coordinate accessed in the matrix. Finally, the process of obtaining the root is based on a calculation of its length, a series of complex transformations, and dictionary look-up. HLAL, 1984: 64–65, and 1987: 194–195, describes a similar process wherein each morphological rule consists of "the triplet $CATP$ (prefix category), $CATS$ (suffix category), LR [Root Length]" and indicates "the series of actions to be undertaken"; prefixes and suffixes are looked up in a glossary and then the length of the assumed root is calculated; finally, in HLAL, ibid. 195, the matrix, or triplet, "($CATP$, $CATS$, LR) gives access to the algorithm which it is appropriate to apply in order to complete the analysis of the word".

The above model is open to criticism in several respects.

First, The use of an affix matrix appears to be imposed by the implementation detailed in BEN HAMADOU, 1986b: 288, in the computer language, FORTRAN; since FORTRAN functions efficiently only with rigid types of data structure, such as arrays and matrices, it is necessary to adopt this type of structure in order to manipulate indexes and access the database.

Second, the validation process, according to HLAL, 1987: 194–195, has to use validation rules, in order to reject ungrammatical analyses, which in the case of function words alone amount to 302 validation rules. Moreover, according to HLAL, ibid. 195, a "root-pattern"

compatibility has to be calculated for further validation of a given analysis, since the pattern is listed in the form of an "operand" which may or may not be valid, and since the root has an "assumed" rather than an actual form, until validation. Similarly, BEN HAMADOU, 1986b: 288, has to reach congruity at three different levels:

- Compatibility between the prefix (P) and the suffix (S).
- Compatibility between the couple (P, S) and the infix (I).
- Compatibility between the triple (P, I, S) and the root (R).

Since compatibility has to be checked at each of the above levels, there is considerable redundancy. In addition, the method, as described in BEN HAMADOU, 1986b: 286–287, and HLAL, 1984: 71, and 1987: 195, relies on combinatorics rather than linguistics, and hence is essentially ad hoc. For instance, HLAL, 1987: 193, reveals that his glossaries contain a set of Arabic prefixes, each of which may recursively contain a subset of prefixes, as in 'wabi:alo$ma', "and-in-the-Nominal prefix", 'bi$:alo$ma', "in-the-Nominal prefix", and 'wabi:alo', "and-in-the", where such prefixes have been redundantly listed, as if they were completely unrelated, and arbitrarily processed, a priori.

A **third** problem with this model is the absence of any basis in linguistic theory in the work done. Thus, all but not only valid analyses are produced: the combinatorial method generates also invalid combinations. Hence, BEN HAMADOU, 1986a: 4, points out that "all combinations are produced initially"; DEBILI and ZOUARI, 1989: 3, confess that they are "contented with obtaining all the possible segmentations upon exiting the MA" which leaves to other eventual analyses (discussed below) the task of rejecting invalid results. The necessity of investigating the invalid combinations, which make up almost two-thirds of BEN HAMADOU's matrix, (or using a large number of rules for HLAL, or using a large number (138) of prestored conjugation tables for DEBILI and ZOUARI), stems from the lack of linguistic (morphological) constraints that could significantly reduce parsing time and means that the analyser has to undertake a large number of decompositions and searches for every word in a text, and will therefore be slow and inefficient.

Furthermore, the analysis in DEBILI and ZOUARI, 1989, is also ad hoc. They argue, ibid. 3, that MA "depends on higher levels", which are defined, loc. cit., as "syntactic, semantic or even pragmatic" and which are discussed in detail, in DEBILI, 1989: 6–11, where further levels (concerned with punctuation, orthography, and lexical collocation) are added. In discussing these levels, DEBILI, loc. cit., draws heavily upon French examples and extrapolates, arbitrarily, their analyses to Arabic morphology. For instance, he tackles, ibid. 8, the question of abbreviated words, such as the French 'sec.', "seconds", then, he gives an example for disambiguating it

from the French, 'sec', "dry". He then generalizes, ibid. 9, that "conditions of a syntactic nature are sometimes needed to effect the right choice". However, DEBILI, 1989, and DEBILI and ZOUARI, 1989, do not explain how and indeed why should a syntactic component do the work of the morphology. The same can be said of the semantic and pragmatic levels. This is in addition to the difficulties inherent in building interfaces between such multilayered approaches to a given analysis.

Moreover, the algorithm used by BEN HAMADOU and described by BEN HAMADOU and SAIDI, 1986: 15, is an ATN-based formalism, using a CFG and proceeding according to a top-down method with backtracking. HLAL, 1987: 195, also recognizes that he has to perform the "annulment" of an analysis "in cases where an attempt at breakdown [decomposition] leads to a negative result". The use of this technique which HLAL, loc. cit., calls "unstacking", amounts to backtracking. Similarly, ALI, 1989: 1–10, suggests an ATN framework for the automatic diacritization of written Arabic. However, his algorithm, loc. cit., is augmented with "a special routine" which the analysis uses to "'undo' the morpho-phonology". Later, ALI, 1989: 9, candidly reveals the nature of this special routine, expounding that his processor uses "intuitive devices driven by heuristics and preferential semantics, and—in few [sic] cases—technical tricks [sic]". Note here that, besides the ad hoc nature of this approach to MA, ALI, 1989, introduces, arbitrarily, the phonological level of analysis, when he claims he is dealing with written text. He defends this attitude, ibid. 2, explaining that the motivation for his program was "successful commercialization" which "has imposed severe constraints on the system and programs [sic] design". ROOCHNIK, 1989: 1, 3, and 4, also proposes an ATN model which is parsed according to a top-down strategy with backtracking. However, he concedes, ibid. 3, that "the subsequent states [to the initial state in his ATN] do multiply in their complexity", this is the familiar problem caused by left-recursion in ATNs.

The choice of the top-down method, adopted by all the above analyses, means that they suffer from the usual difficulties associated with ATNs, namely the computational inefficiency and the expense which are caused by backtracking in ATNs. Such difficulties are a consequence of the production of all possible analyses and the non-early rejection of invalid results. For instance, ROOCHNIK, 1989, describes a fairly inefficient system that is based on a simplistic view of Arabic morphology. In his system, he resorts to preautomatic, a priori, and ad hoc segmentation of Arabic words, as in his example, ibid. 2 and 12, 'al kariimu', "the generous", where 'al', "the", normally attached to the word in Arabic, is artificially and arbitrarily separated from the rest of the word (cf. our discussion of SM in Ch. 1, § I.3.1). ROOCHNIK's analysis merely illustrates the "strip-off-the-endings" approach to morphology: his program, ibid. 5, "strips the suffix off of the current word [...], matches the suffix against a list of endings in the lexicon, and matches the stripped [word] stem against a list of same". Thus, MA is reduced to suffix stripping and "stem" recognition. ROOCHNIK, ibid. 6, gives the basic form 'katab', "write",

as an example of what he means by stem. This means that, no work is done to identify roots or patterns within the basic form identified.

II.2.2 DCG-Based Analysis of Arabic Morphology

1) MEHDI, 1985 and 1986, describes a DCG-based algorithm for the syntactic parsing of Arabic which includes a word analyser as well as some semantic work. MEHDI's formalism, is based on work by PEREIRA and WARREN, 1980, implemented in PROLOG, and run on a PYRAMID 90X computer. MEHDI, 1985 and 1986, has very little detail on the kind of MA done or the formalism used to do this work, even though he stresses that the word analyser "is the most important part in the system" (MEHDI, 1985: 15). However, he explains, ibid. 17, that word recognition is performed by stripping a word of its affixes and looking it up in a "root-oriented" dictionary. He also specifies, 1985: 12, and 1986: 12 and 17, that the parsing proceeds in a "top-down" and left-to-right" manner and that his system uses backtracking in case of failure. However, he admits, ibid. 12, that this strategy "is not the most efficient one". Thus, his method, besides performing essentially ad hoc MA, is constrained by the implementation in PROLOG and the choice of the inflexible and inefficient top-down backtracking strategy.

2) IBRAHIM et al., 1989: 1–12, also describe a DCG-based algorithm for an English-Arabic MT project. They give brief details, ibid. 6, about the proposed MA, which is based on work by PEREIRA and WARREN, 1980, and implemented in PROLOG. It is suggested by IBRAHIM et al., loc. cit., that

> the logic programming is used to represent this [morphological] knowledge in terms of facts and rules. The process for isolation and recognition of morphemes is described in terms of the predicates: root, weight, strip, and strip-reverse. The strip and strip-reverse predicates are used to recognise and isolate, in a recursive way, the antefix, prefix, suffix and postfix letters. The root and weight predicates are used to extract the root letters and deduce all possible morpheme schemes.

3) ELSADANY and HASHISH, 1989: 600–612, describe an Arabic morphological system, which is implemented on an IBM PS/2 Model 60 at the IBM Cairo Scientific Center and which was written in PROLOG. However, they note, ibid. 611, "work is still in progress to achieve similar results [to those achieved for the verbs] for the Arabic nouns". ELSADANY and HASHISH, ibid. 600, start with a strong attack on the use of "systems developed for the European languages [which] are not convenient for the use of Arabic because of the nature of the language and its writing system". They then proceed to describe a morphological system of which the three main components are developed for European languages.

First, they point out, loc. cit., that it is "formulated using the conventional generative

grammar". (In particular, they refer, ibid. 604, to ANDERSON's 1982 theory and subscribe to his position of making the syntax responsible for inflectional morphology.)

Second, they declare, ibid. 607, that their program is written "using logic programming language" as specified in PEREIRA and SHIEBER, 1987.

Third, the program unconventionally (and in unprecedented fashion), combines on the one hand, the PROLOG DCG mechanisms, which is used mainly for derivations, with, on the other hand, an ATN formalism, which is used, according to ELSADANY and HASHISH, loc. cit., for "the removal of the affixes and the extraction of the stem and the morphological features".

This last choice is tantamount to a claim that Arabic morphology is characterized by a dichotomy of two irreconcilable sets of phenomena and which are to be accounted for by two different formalisms that are as much at variance as a logic DCG formalism and an ATN model. Such a fantastic claim has never, to our knowledge, been made for any language. In any case, ELSADANY and HASHISH offer no theoretical justifications for such a claim, nor do they offer any evidence for correspondence or affinity between Arabic morphological structures and the formalisms they chose. They do not justify the rigidity of these formalisms and their PROLOG implementation.

Moreover, faced with the discrepancies between certain morphological deep structures and surface structures, ELSADANY and HASHISH, ibid. 609, simply state that "artificial [essentially ad hoc] measures [i.e., patterns] are created for the different surface realizations of Arabic nouns" on the grounds that "the problem faced was that the rules governing morphological changes are not well defined and are not complete in the Arabic literature".

II.2.3 Two-Level Finite-State Analysis of Arabic Morphology

BEESLEY, 1990: 1–5, and BEESLEY et al., 1989: 1–10, describe a "Two-Level Finite-State Analysis of Arabic Morphology" which is based on KOSKENNIEMI's Two-Level Morphology. The formalism they describe, ibid. 9, is written in the computer language *C* and run on an IBM 4361 mainframe. BEESLEY et al., ibid. 4, claim that "at no time does the system resort to ad-hoc 'cut-and-paste' methods to reduce surface words down to a stem or root that can then be looked up. All Arabic constructions, including interdigitated stems, are found by following paths through [abstract] lexical trees.". However, they concede, ibid. 1, that "to handle Arabic and other Semitic languages, we found it necessary to enhance the traditional two-level implementation design with 'detouring' to allow proper analysis of stems, where roots and patterns interdigitate". Similar reservations are expressed by BEESLEY, 1990: 2.

Since, this model posits the existence of an abstract lexical level on which the analysis depends in order to recognize morphological surface forms, it is necessary for the program to guess the form of non-realized Weak letters in order to match the abstract representation (BEESLEY et al., ibid. 5). Further, the use of this abstract level means, as acknowledged by BEESLEY et al., ibid. 9, that their program has problems recognizing Doubled roots. Thus, as we have argued (in § I.2.5 above), the use of an abstract level adds great complexity to KOSKENNIEMI's formalism in a similar way to the complexity of a generative transformational theory. Hence, BEESLEY et al, ibid. 3, point out that "dozens of such [extra] rules may be necessary to correctly relate the lexical and surface levels". Similarly, BEESLEY, 1990: 4, confesses that "the large number of rules [in the system] reflects the obvious fact that there are remarkable discrepancies between the lexical level and [Arabic] traditional surface orthography", that "the huge number of possible zero realizations in Arabic proved a significant overhead that hurt performance", and that, even after all this, "the system finds an average of five morphological solutions for each surface word [...] many [of which are] highly unlikely solutions".

II.2.4 Other Recent Analyses of Arabic Morphology

1) ALBAWWAAB et al., 1989: 25–63, describe a project for the automatic generation of Arabic derived forms which is to be implemented on a VAX 11 machine. The project is only half completed and handles only the generation of derived forms from roots and words, but is currently unable to recognize actual complex words and extract roots from them. In addition, this system can only generate *simple* forms (root and pattern realization plus the affixes immediately attached to it) and does not account for antefixes and other affixes.

The method followed by ALBAWWAAB et al., 1989: 25–63, is based on a statistical combinatorial approach to derivation, which obtains "all possible Augmented forms of the Basic Verb be it Triliteral or Quadriliteral, and whether this Augmented form actually exists in Arabic dictionaries or not" (p. 34). Thus, we find several examples, (pp. 36 to 49), which are caused by the adopted principle of generating all possible combinations and derivations, regardless of their grammaticality. For instance, ALBAWWAAB et al., 1989: 34, give the examples of the derived Verb forms, '*:ikotatabba', '*:ikotabba', and '*takattaba' which do not exist and which are generated because of the reluctance of the authors to use linguistic constraints which will generate "all and only" grammatical forms.

2) BRUSSET and ABDELGHANI, 1989: 9–28, describe a project called *SYAMSA* for the morphosyntactic analysis of Arabic lexical databases. However, their system is still "currently under development" (p. 24) and is presently "operational for a very limited lexicon" (p. 9). As far as the design of their system is concerned, they confess, loc. cit., that the intended implementation "on an IBM-PC compatible Microcomputer" has imposed certain constraints,

in terms of "response time" and "memory optimization".

3) ARISTAR, 1987: 67–75, describes an Arabic morphological analyser. From personal communication in 1986, it emerged that this analyzer, which is called *NABU* and is part of a multilingual project started in 1984, is implemented on a Symbolics machine and written in *Zeta LISP*. However, it is not clear, from ARISTAR, 1987, if and whether the project is yet completed. The NABU analyser, according to ARISTAR, ibid. 74, is characterized by early rejection of invalid analyses, which is a good advantage in parsing. ARISTAR, ibid. 71–72, shows that his parser is based on a *unification* grammar (*unification* is defined by PEREIRA and SHIEBER, 1987: 18 and 47, as a matching process wherein "two atomic formulas unify if there exists an assignment to the variables in them under which the two formulas become identical").

The unification technique described by ARISTAR, loc. cit., essentially attempts to match initial hypothetical graphs generated by the parser by actual analyses performed for corresponding parts of the input, which is segmented in a left-to-right direction. However, the choice of this direction is unfortunate and is not very well suited to Arabic morphological structures. We shall argue, in this thesis, that it is more appropriate to follow a context-sensitive method in the analysis of Arabic structures. This means proceeding in a right-to-left direction whenever possible but also using left-to-right parsing where appropriate on the grounds that there are fewer affixes on the right of a word (cf. § III.2.3.1 below) and because of the long-distance interdependencies which exist between enclitic affixes and proclitic ones, for instance, if a pronominal suffix is found, it would be absurd to look for a prefixed Determiner (cf. Ch. 6, § II.1.6). In ARISTAR's case, the choice of the left-to-right method is forced by the unification technique, which, because of its rigid mechanism, cannot suspend parsing in one direction and consider the context of the whole structure by looking at proclitic structures.

4) In addition to the above analyses, other attempts (such as BEJAOUI, 1985; BOUBAKER, 1986; LAZREK, 1986; and MOURTAKI, 1986) have been made to analyse Arabic morphology by computer using *expert systems* (an *expert system* is defined by FORSYTH and RADA, 1986: 256, as a computer program that is "capable of offering advice or making intelligent decisions in a relatively narrow subject area" and that contains "an *Inference Engine*, which is a set of reasoning methods, and a *Knowledge Base*, which stores the system's expertise"). Such attempts are irrelevant to the present framework, since they adopt a cognitive rather than a linguistic approach: their starting-point is the mapping, in the form of computational models, of what humans do when they solve a problem (cf. FORD, 1987: 73), as opposed to the formal analysis of morphosyntactic data.

II.3 A CRITICAL SYNTHESIS OF PREVIOUS ANALYSES

Above, we have made a number of criticisms of individual computational analyses. However, in order to get a more broad picture of the general trends of previous work, there are a number of critical points that are best made collectively about it. We note that most early computational work was adversely affected by the computer technology available in the 1960s and early 1970s and is now dated by the advances in this field. For instance, BATHURST's project was restricted in its implementation: he admits, ibid. 188, that the keyboard character set of the machine was limited, that there was "a limitation on punctuation marks, and an inability to show short vowels, even when given in print", and that "some compromises had to be made". SATTERTHWAIT, 1963: 63–66, BATHURST, 1971: 188, SCHMIDT, 1971: 12, and FRAZIER, 1976: Vol. I, 66, describe programs which were implemented using "key-punching" of texts "on paper tape". Similarly, SATTERTHWAIT, 1963, used IBM 709 or 7090 machines, BATHURST, 1971, used a Cambridge Titan, and BECKER-MAKKAI, 1970, used an IBM 7040/7044.

In addition, many previous analyses require user intervention in order to work. Thus, ELSADANY and HASHISH, 1989, avowedly, rely on extra information to be "given by the user", and need a "decision module" which sorts out invalid results amongst the overgenerated set of all possible results (cf. pp. 610–611). Similarly, BRUSSET and ABDELGHANI, 1989: 9, depend on specifications by the user for the choice of a grammar type which can be either a top-down method or a bottom-up one (p. 12). The user also has "to change or correct words that block the analyser" (p. 13). Furthermore, many previous analyses lack any details about the implementation of the work described. Thus, COHEN, 1961, does not reveal all the details of his work's implementation; ALI, 1989, and IBRAHIM et al., 1989, do not give full details on the parsing strategy they adopted for their work or indeed on the extent to which their projects have been actually completed; and ALI, 1989, does not specify the machine and the computer language used in his work.

More seriously, most previous analyses fail to meet observational adequacy in many respects and present a simplistic incomplete view of Arabic morphology which fails to take account of the complexity of Arabic morphological structures (cf. our analysis in Ch. 10, § I.1.1). Thus, BECKER-MAKKAI, 1970: 927–931, like SCHMIDT, 1971, FRAZIER, 1976, and many others, limits her computational analysis to Arabic Verbs and does not take account of Nominals, pronominal suffixes, the Question Particle, and other prefixes; SATTERTHWAIT, 1962: 7, 89, and 297, arbitrarily excludes from his Arabic words, duals, perfect-tense forms, jussive and subjunctive forms of the Verb, passives, as well as Nominals with pronominal suffixes; FRAZIER, 1976, and SATTERTHWAIT, 1962, 1963, and 1965, do not analyse Arabic antefixes, long-distance correlations between non-adjacent affixes, nor any complex morphological structures; ELSADANY and

HASHISH, 1989, ignore the Question Particle ':a', as well as the Future Particles 'sa', "going to" and 'li', "in order to"; similarly, BRUSSET and ABDELGHANI, 1989, do not analyse the Question Particle and pronominal suffixes; and finally ARISTAR, 1987, simply ignores the Question Particle and prefixed Conjunctions. Having dispensed with such complexity, no ambiguities need to be solved and it becomes relatively simple to perform the Arabic conjugations and derivations which they describe.

Even more seriously, most previous analyses lack any theoretical justification which relates the choices made in terms of linguistic analysis and computational techniques to the properties of the Arabic language. Thus, BECKER-MAKKAI, 1970, proposes a method for Arabic Verb derivation, which is based on *IC* analysis, but she shows no theoretical correspondence between it and the structure of Arabic words; similarly, BEN HAMADOU, 1986a, and 1986b, BEN HAMADOU and SAIDI, 1986; DEBILI, 1989, SAIDI, 1985, DEBILI and ZOUARI, 1989, and HLAL, 1984 and 1987, fail to explain their choice of an ATN method; IBRAHIM et al., and MEHDI, 1985 and 1986, fail to justify their choice of the DCG formalism, (which, we have seen, is a rigid one, since it imposes a strict logical representation for language objects in which they are not readily expressible), as well as failing to justify the choice of an implementation as rigid as the PROLOG one; FRAZIER, 1976, BRUSSET and ABDELGHANI, 1989, and ALBAWWAAB et al., 1989, offer no theoretical justifications for their analyses; while BEESLEY et al, 1989, and BEESLEY, 1990, ignore the reservations cited above and made by KOSKENNIEMI himself, 1983: 27, about the adequacy of his system for languages with infixation and reduplication, such as Arabic. Thus, they do not relate the linguistic features of the Arabic morphological system to KOSKENNIEMI's formalism. Moreover, some analyses (such as ROOCHNIK, 1989) naïvely maintain the "strip-off-the-endings" approach to MA in the face of the complexity of the Arabic morphological system. Hence, proposals such as those of ROOCHNIK, 1989, and ALI, 1989, can only be theoretically uninteresting, uninformative, and futile and would yield very little in the way of positive results for Arabic words which display any small degree of morphological complexity.

Finally, previous analyses of Arabic morphology by computer have suffered mostly from their insistence on the combinatorial approach to language analysis, which, because it produces all possible (grammatical and non-grammatical) results, is essentially inefficient and fails to formulate constraints which favour the early rejection of the invalid results. In particular, COHEN, 1961, BATHURST, 1971, SCHMIDT, 1971, FRAZIER, 1976, BEN HAMADOU, 1986a and 1986b, DEBILI and ZOUARI, 1989, and ALBAWWAAB et al., 1989, suffer from the consequences of adopting the combinatorial approach to MA which is not surprising, given that most of the work reviewed here has been done by engineers and mathematicians rather than computational linguists.

III TUNIS1: A MODIFIED SEARCH METHOD FOR A MORPHOLOGICAL ANALYSER OF ARABIC

In this thesis, we propose a more efficient approach to the morphological analysis of written Arabic. We claim that our Arabic morphological analyser (which is implemented as a program called *TUNIS1*) has considerable advantages over the above techniques in terms of speed, economy of secondary storage, database access, and theoretical soundness (in particular, it does not rely heavily on combinatorial mathematics, but adopts an algorithmic approach based on theoretical linguistic principles).

In addition, TUNIS1 is implemented in Cambridge LISP, which is altogether more appropriate than languages such as FORTRAN, PASCAL or PROLOG for this type of endeavour. TUNIS1 is also based on *search strategies*. In order to justify the validity of our approach, we describe below what *searching* is and why it is desirable as a computational technique and then we proceed to describe the analyser itself.

III.1 SEARCHING AND DEVISING A SEARCH SCHEME

Some problems are best solved by considering them as offering a set of answers to be looked through for the right solution. This is known as *searching*, which means "to scan a set of data items in order to locate those having a given property" (cf. DARCY and BOSTON, 1983: 223). The number of possibilities to consider is called a *search space*. In order to look up an answer in a search space, there is a variety of available techniques (called *search techniques* and ranging from *blind exhaustive search* to *directed search*) which aim to optimize searching. The optimization of a search involves minimizing its *cost*. For instance, if a search problem is characterized by *combinatorial explosion* or exponential increase of the routes that lead to a solution, an optimal strategy is to reduce the size of the search space by *directing* or *guiding* the search, for example by looking for the expected case first.

Each search problem has an *initial state* where the search starts from, and a *goal state description* which allows us to recognize the right solution. Each state generates further states, said to be its *descendants, successors,* or *children,* which it is hoped lead closer to the goal. The set of all states is usually represented as a *tree*, with the *nodes* being the generated states, and the *branches* being the paths. The search space can be examined by defining a search algorithm (if the space is too big to warrant its exhaustive enumeration) or by adopting one of a number of search strategies (if the space is so small that it is viable to adopt a brute-force systematic search). Such systematic search methods are of two basic types: *depth-first* and *breadth-first*.

1) In *depth-first search*, only one of the children of the starting state S is visited. Then the

goal state is looked for among its descendants always moving down the tree. If a childless node is reached, then the searcher backtracks to the last point where a choice was made and develops its children. One problem with depth-first search is that if there is an infinite number of states reachable from the initial state, the search space can increase exponentially in size. This problem can be solved by a technique called *depth cut-off* which imposes a maximum number of deep states to be tried before moving back up and down the tree again. Another problem with the depth-first method is that the searcher may walk blindly through repeated states without realizing that they had been visited before. This may lead to extensive redundancy. Thus, depth-first search is best suited for finite search space problems with a minimum of repeated states.

2) In *breadth-first search*, progress through the search tree is made on a level-by-level basis, all the nodes on each level being investigated before moving down to the next level. This means that every state will eventually be examined without the need for a depth cut-off. A breadth-first search technique can be useful with big search space problems as it does not risk getting lost and never returning back and it does not need to backtrack or look at repeated states.

In addition to the above kinds of search, there are special cases of search problems where a search goal can be satisfied by solving *subgoals* and each subgoal constitutes a search problem in its own right. This is the case of *Goal Trees*, or *G-Trees*. Each *G-Tree* is a set of subtrees which can be *solved* if the initial node is solved, that is if it does not need any further consideration. The process of solving tree nodes is referred to as *pruning* a tree. Several methods may have to be tried before one satisfactory subgoal, or *success* node, is found. The nodes of such G-Trees are of two types:

- *and*, or *obligatory*, nodes which must all be satisfied; and
- *or*, or *choice*, nodes of which only one need be satisfied.

and nodes can be viewed as *plans*, or obligatory things to do in order to reach the goal, and *or* nodes as *options*, or choices. A variant of G-Trees is one where *and* nodes have a constraint attached to them, which is applied if all the children are solved. So the *and* subgoal obtains if it and all its children are solved and the constraint holds. Constraints can be imposed at different points of the search, after all the children of an *and* node are solved or beforehand if a violation of the constraint can be detected earlier. Hence, the set of *and-or* nodes in a G-Tree allows a natural progression towards search strategies as well as an appropriate means of expressing our morphological CSG (cf. Ch. 1, § II.3.2). Hence, we propose that G-Trees account adequately for the Arabic linguistic data by allowing the direct representation of MS rules as nodes, and of morphological conditions as constraints on the satisfaction of a given node. In order to use

G-Trees efficiently in parsing and in particular in order to devise an optimal strategy for word recognition, we just have to translate M-Trees to G-Trees. Such a translation can be achieved using the following formal Conversion Rule:

$\forall$ *M-Tree*, *if* $X \longrightarrow (A) \quad B \Longrightarrow$

G-Tree, $X \longrightarrow Y \quad Z$; $Y \longrightarrow B$; and $Z \longrightarrow A \quad B$.

Having described the theoretical background to our computational analysis, we can now proceed to describe the morphological analyser itself.

III.2 MORPHOLOGY: THE PROBLEM

Arabic morphology shows complexity at two levels:

1) Morphemes contain a number of bound morphemes which are attached in Arabic writing to a core word. The core word itself is an amalgamation of a *centre* and affixes. Each centre is formed through the combination of a root with a pattern (e.g.,
'drs' and 'cvcvc' $\rightarrowtail$ 'daras', "study").

2) The affixes (prefixes, infixes, and suffixes) added to the centre indicate things like NUMBER, GENDER, PERSON, MOOD, CASE, TENSE, VOICE, etc., (e.g.,
'daras' and 'uw|' $\rightarrowtail$ 'darasuw|', "they studied").

Furthermore, a core word may undergo the addition of other affixes, such as *proclitic antefixes* and *enclitic Pronouns*. Such combinations often give rise to structural *ambiguity* (we define an *ambiguity*, after CHARNIAK and McDERMOTT, 1985: 197, as a place "where more than one rule may apply". For example, 'd-ahaba', may be a Verb: "went", or a Nominal: "the gold of"; 'mudarrisiy', may be a complex Nominal: "my teacher", or a simple Nominal: "(the) teachers (of)"). Moreover, the various segments have an almost infinite potential for combining with each other, restrictions on possible combinations being imposed by grammatical conditions. The above problems create non-trivial difficulties for computational analysis. Identifying the component segments of a given word is rendered even more problematic because their combinations often give rise to graphemic modifications which cause the morphological structure of the word to be opaque. We propose an alternative computational solution to the problem, which is constrained by a linguistic theory of morphological structure (also described briefly by DEGACHI and SMITH, 1990).

III.2.1 Tasks of the MA of Arabic

We need to carry out specialized tasks in order to be able to process Arabic morphology by Computer:

1) Discover the graphemic structure of the various component segments of a word (by identifying the boundaries of each segment) and discover their morphosyntactic CATEGORIES and which of them (e.g., Nominals, Verbs) are valid for the centre of a given word.

2) Enforce morphological rules of concatenation and discover the precedence relationships, correlations, and interdependencies, that hold between adjacent segments. (We noted above that the various word segments have an almost infinite potential for combining with each other but that there are restrictions on possible combinations. For instance, the combinations are not reflexive, in that segments do not combine with themselves (e.g., '*bi$bi', "*in in"), and are not symmetrical, (so, for e.g., 'wa$bi', "and in", but not '*bi$wa', "*in and").

3) Discover the possible LENGTH intervals for each of the segments of a word and whether the lengths of roots and bound morphemes can serve as constraints.

4) Give some clue as to reason, such as an error message, when a word is rejected as invalid.

5) Achieve the morphological parsing of a long complex sentence in a reasonable timing objective of less than one minute in order to be on target for viable timing in syntactic processing, although this has to be a secondary objective compared to the primary objective of the analysis, which is to demonstrate the validity and theoretical soundness of our linguistic approach.

III.2.2 Components of the Analyser

III.2.2.1 MS Rules and Constraints on Rules

The morphological analyser, TUNIS1, incorporates the morphological CSG (described in Ch.1, § II.3.2) which specifies a set of MS rules that adequately describe linear precedence, express affixation constraints, and provide an exhaustive definition of Arabic word structure. The MS rules, which serve as the basis for the computational procedures, are translated into G-Trees which are the input to the search strategy and the parsing algorithm: the TUNIS1 algorithm is a modified cut-off search algorithm which uses abstract objects called *G-Trees*; (we have argued above that G-Trees, as *or* and *and* nodes, allow a natural progression towards search strategies). The affixation constraints are then placed as exit conditions on node satisfaction, which allows TUNIS1 to filter out invalid parses as early as possible, rather than attempting to generate all possible combinations. Such conditions can relate

to the precedence relations (as above),

to the form of segments (certain segments may assume only a very limited number of forms, e.g., the antefixed Question Particle, in an Arabic word, can only be ':a', meaning "is ...?"),

to the LENGTH of segments (each segment has a specified character-length interval, ranging between a minimum value (the *threshold*) and a maximum value (the *upperbound*), and

to CASE and MOOD conditions (the occurrence of particular forms of Nominals and Verbs is constrained by the kind of Particle occurring in prefix positions in the word, e.g., the occurrence of 'bi', "at, in, ...", or 'ka', "as", requires a Nominal in the oblique CASE while the occurrence of 'sa', "going to" requires a Verb in the indicative MOOD).

III.2.2.2 The Database

The MA tasks also involve the discovery of optimal and cost-efficient methods for the storage of the data structures in order to optimize data access and data manipulation. Our morphological grammar allows us to organize the morphological data into a structured *database* (which is a superset of dictionaries) so that its design optimizes our search strategy. Entries in the database have the form of simple LISP-list structures and each entry includes an *identifier* with an associated *value*. The value is a list of *atoms* each of which in turn has a set of *properties* associated with it. The entries are grouped into several classes representing lexical entries for affixes, roots, patterns, clitic PERSON MARKERS, CASE and MOOD MARKERS, and suffix Pronouns, as well as the vowels, semivowels, and consonants of Arabic.

III.2.3 The Implementation of TUNIS1

The implementation of TUNIS1 involves two main tasks: computational analysis and linguistic analysis.

III.2.3.1 Computational Analysis

Since our parser needs to identify the component segments of a given word, TUNIS1 must incorporate a general abstract SEGMENTation, or decomposition, procedure which can be called as a subroutine from any point in the program. This subroutine can make use of arguments such as the input, the LENGTH (L) of the input, and a numerical value, which corresponds to the required number of characters in a segment. In order to identify the segments returned by the SEGMENTation procedure, TUNIS1 must first search for them in the database. The search strategy adopted by TUNIS1 is based on a modified depth cut-off method rather than a breadth-first one. However, top-down and depth-first methods are traditionally associated with backtracking. TUNIS1 does not need to backtrack, since it proceeds in an exhaustive manner and considers all possibilities, including ambiguous ones, in an ordered way. This exhaustive method follows from our objective of imposing conditions on exits.

However, what is the optimal path to follow in this search? The decision on the optimal solution of a G-Tree depends on both practical and theoretical issues and can be inspired by two basic principles. From a linguistic point of view, the minimal valid concatenations are intuitively likely to be most frequent, on the grounds that the occurrence of longer concatenations is going to be restricted by the principle of "the least effort" that underlies language use. Hence, our parser should start with the simplest possibilities and progress towards more complex ones. The other issue is in which direction should we proceed? We note that, while there are three optional prefixes at the front of a word (which are: Q(uestion Particle), C(oordinate Conjunction), and P(refixed Particle or Preposition), and which we refer to in Chapters 10 and 11), there is only one optional affix which may end a word and which is an enclitic Pronoun. Therefore, our parser should begin by proceeding in a right-to-left direction whenever that is not in contradiction with the first principle. Thus, we can state our parsing philosophy as taking the simplest, most likely route first, and then proceeding in a right-to-left direction first.

In addition to SEGMENTation and search, TUNIS1 needs to incorporate DERIVation procedures which can handle the recognition of Verb and Nominal centres of words. Since the recognition of the whole word will depend on it, DERIVation is a central task in the algorithm; it involves two subroutines:

1) Root identification which needs to extract the root in the inputword by reducing its centre to purely consonantal clusters to be looked up in the database. If a root is not identified the algorithm has to consider problems caused by modifications, (for e.g., at the boundaries of segments), and to solve them by applying operations, such as deletion and substitution, at linguistically predetermined positions in the word.

2) Pattern MATCHing which needs to recognize three types of data: consonants, vowels, and semivowels. The task of the pattern MATCHer is to retrieve a list of the patterns associated with the obtained root and stored in LISP-association lists in the database. It then has to scan the list of patterns comparing it with the centre of the inputword and trying to MATCH its vowels and semivowels with their exact counterparts in the lexicon and its consonants with consonant slots in the pattern.

III.2.3.2 Linguistic Analysis

The successful recognition of a word should trigger the ASSIGNment to it of the linguistic property values derived from its structure during the analysis. The ASSIGNment of such values should first make use of the default values ASSIGNed in the database to each morpheme and for each of the linguistic properties (listed in Ch. 1, § II.3.1) but it should also be sensitive to the structural morphological context provided by the root, the pattern, and all the affixes. The parser should also be able to scrutinize contexts in order to resolve structural ambiguities

resulting from variation and ambiguities of property values. Furthermore, the computational analysis of an Arabic word should also culminate in ASSIGNing to it a structural morphological description, and producing this description in the form of a parse tree. Both this parse tree and the list of ASSIGNed linguistic properties should be made available to the user as the output of the parser.

We start the detailed account of the computational and linguistic analyses of TUNIS1, in Chapter 3 below.

==0==<*****>==0==

Chapter 3

PRACTICAL ISSUES IN COMPUTATION

INTRODUCTION

In Chapter 3, we provide an account of some practical issues in computation. Specifically, we are concerned with issues in the choice of a programming language, and with some processing generalities in morphological parsing.

In Section I, we give a brief description of the *LISP* language and, in particular, the Cambridge LISP version. We provide a description of programming methods, function definition, data representation, data access, and data manipulation, as well as property ASSIGNment in LISP. It is hoped that, by explaining the advantages of LISP programming, this Section will provide a justification for the choice of LISP as the appropriate language for our morphological program.

In Section II, we deal with some generalities of morphological parsing. We give a brief description of some general independent operations, which are called *closed subroutines*, and which we can define in terms of the functions and procedures described in Section I. These subroutines, which are necessary for the operation of our morphological parser, have to specify algorithmic methods for performing tasks such as SEGMENTation, MATCHing, character PICKing, word ASSEMBLY, and TRANSCRIPTION, as well as operations such as DELETion, SUBSTITUTion, and COPYing.

I PARSING IN CAMBRIDGE LISP: WHY LISP?

In this Section, we want to give a flavour of the LISP language but not an in-depth introduction. *LISP*, an acronym for LISt Processing, is a high-level programming language invented by John McCARTHY in 1960. Although it can be used for mathematical calculations, LISP is a problem-orientated functional language designed for list processing and symbol manipulation. Since the 1960's, LISP has continued to evolve, with the development of several dialects such as *MACLISP*, *INTERLISP*, *Common LISP*, *Standard LISP*, and *Cambridge LISP*. This last one is the dialect we use here. It is written by FITCH (of the School of Mathematics at Bath University) and NORMAN, 1977 and 1985. The version we use is one known as *Version 1.05 of Cambridge Orion LISP* (hereinafter CL, or LISP) and is normally run in about one Megabyte of RAM. This version is available at Bath University on a Main Frame HLH Orion machine with 8 Megabytes of RAM and 600 Megabytes of hard-disc memory, and which uses an INTERGRAPH *Fairchild-Clipper 32-Bit Microprocessor*, a fast processor with most instructions operating at one thirty-nanosecond-clock cycle (one nanosecond being one thousand millionth of a second).

LISP has been gaining wider and wider use across the spectrum of computer applications. WINSTON and HORN, 1981: 3, remark that "these days, there is a growing armamentarium of programs that exhibit what most people consider intelligent behavior. Nearly all of these intelligent or seemingly intelligent programs are written in LISP.". They cite, loc. cit., as examples of such programs expert problem solvers, common sense reasoning, knowledge representation, and learning. GLOESS, 1982: 5, notes that LISP "is now the most widely used programming language in the Artificial Intelligence community". CHARNIAK and McDERMOTT, 1985: 33, note that "there are occasional programs in AI written in languages other than LISP, but if one were to take, say, the 100 best known programs in the field, probably 95 would be in LISP, and of the remaining five four were written before LISP was available on most computers".

The success of LISP in Artificial Intelligence work is due to many reasons. Languages like FORTRAN were, according to LEHMANN, 1978: 154, "developed for the purpose of numerical manipulation", whereas LISP is geared to handle many different types of complex data structures such as lists and symbols, called *ATOMS*. Hence, FARGHALY and BROWNFIELD, 1985: Vol. I, 15, remark that, in LISP, "the notions of the atom and the list correspond very closely to the structure of natural language [...]. This affinity between natural language and LISP as a programming language makes LISP an ideal programming language for natural language processing". LISP is particularly useful for data with variable structure and provides easy methods for storing such structures, ASSIGNing to them, modifying and updating them, inserting, COPYing, and DELETing or simply locating items in these structures. The flexibility of LISP lies in its ability to treat programs as data and provide for the self modification of a given program as it is executing. LISP provides *system-defined* functions which can be incor-

porated into *user-defined* programs or modified at will. The FUNCTIONAL feature of LISP means that, although, it can be used *iteratively*, it provides a natural *recursive* mechanism that is more geared towards optimum problem solving. The recursive feature of LISP turns out to be a felicitous one for Natural Language analysis as recursion is a feature of Natural Language itself. (For further discussion of the terms *functional* and *recursive* see § I.1, below).

I.1 PROGRAMMING IN LISP

LISP is a highly interactive programming language that is both compiled and interpreted. Since interpretable programs are not always compilable but the opposite is true, this is a helpful feature in that the interpreter can be used for error detection and the experimentation and testing of such programs until compilable status is reached.

Although LISP is primarily a *functional* Language, it also provides facilities for *procedural* programming. The procedural, or Von Neumann, approach to computing is to state explicitly a sequential set of instructions to be executed step by step. These steps modify the value of a number of secondary variables which then enables the program to calculate the desired value of one main variable. The *functional*, or *problem-orientated*, approach to programming is to break down each function, in a similar way to mathematical functions, into smaller functions. The value RETURNed by one complex function depends on the value RETURNed by the smaller functions that it invokes in the execution of its task. The transfer of control from one function to another may be done in several ways. The most important feature of LISP is one such method of control transfer called *recursion*. This is also the optimum method for writing programs in LISP. A *recursive* program is one which transfers control to a subprogram which recalls the main program itself. The main program specifies a simple base case and a task to be performed for that case. The input is gradually broken down into smaller units, with the program recalling itself and applying the base case each time the input is modified. When the input is exhausted the value RETURNed is the accumulated value of applying the main program to the whole input.

It is also possible to program in a *procedural* or *iterative* way in LISP, typically using a function called *PROG* to build loops. PROG uses a number of *LOCAL VARIABLES* and starts with a stop or exit condition followed by a number of steps to be evaluated sequentially. The last step is an instruction called a *GOTO* statement, to go to the top exit test and repeat the procedure. The main advantage of *iterative* programs over *recursive* ones is that loops do not use stack memory while recursion does.

I.2 FUNCTION DEFINITION IN LISP

The normal way to define user functions in Cambridge LISP is to use the special form *DE*, which takes a function name, a list of arguments that may be constants, variables or function calls, and a body of instructions. The general form is:

(DE *FUNCTION-NAME* (argument_1 argument_2 argument_n) (body)).

It is also possible to define other special functions such as *MACROS* which are faster and substitute simple and frequently-used chunks of program. They have the general form:

(DM *MACRO-NAME* (argument_1 argument_2 argument_n) '(body)).

Inside the body of a function the user can specify iterations, recursions or simply embed further functions. The LISP interpreter evaluates embedded functions from the inside out, the innermost function being evaluated first, and sequential functions in a strict left-to-right order.

Besides numerical and data manipulation functions described below, CL also provides other special forms such as *NULL* which checks if its argument has a NIL value and *COND* which is a generalized form of the "*if ...then ...else*" form of other artificial languages. COND has the syntax:

(COND
($predicate_1$ consequence_1)
($predicate_2$ consequence_2)
($predicate_i$ consequence_i)
(T consequence_j)).

The above form has peculiar semantics in that consequence_1 is RETURNed only if $predicate_1$ has a non-NIL value. Otherwise $predicate_2$ is evaluated, until $predicate_i$ is reached and if all predicates evaluate to NIL then consequence_j is RETURNed. This form also includes the special identifier T for True. Note that there are not separate values in LISP for Boolean values, the identifier *NIL* is treated as the value *FALSE* and any other value is considered to be TRUE. CL provides special functions to handle the Boolean Operators: *AND*, *OR*, and *NOT*. The arguments of these functions are evaluated in a strict left-to-right order, and not all arguments may be evaluated. *NOT* is synonymous with NULL and takes one argument. *AND* takes two or more arguments and RETURNS the value of the last argument, if all the arguments have a non-NIL value. But if one of its arguments evaluates to NIL, it RETURNS that value as soon as it reaches that argument without bothering to evaluate the rest of the arguments. *OR* takes two or more arguments but does not evaluate all its arguments. As soon as it finds a non-NIL value it RETURNS it as the value regardless of the rest of the arguments.

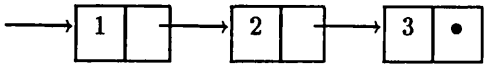
There are also other helpful auxiliary system-defined functions in CL which provide support

for the user. For instance, we can use the function *TRACE* to delineate the detailed process of a program. This can be helpful for simple illustration of the program steps or for error detection, or *debugging*. There are other functions such as *TIME* and *PRETTYPRINT* which supply respectively information about processing time and the *logical* organization of a program. (We use the term *logical* to describe what HANKS et al., 1986: 903, refer to as "the relationship and interdependence of a series of events, facts, etc.". Thus, we will use *logical* to characterize the systematic aspect of the interaction between the component parts of models, or systems, but not in any other technical or cognitive sense that might be used by other scholars.)

I.3 DATA REPRESENTATION IN CL

Cambridge LISP provides *direct-Access-Storage* devices to store structures or programs in virtual memory. Such devices use key sequences, or *identifiers*, to store or access these data. The basic record type in CL is the *LIST* structure which can be constructed from individual objects called *ATOMS* or from other lists. *ATOM* types include integers of arbitrary magnitude, rational and other numbers, vectors, strings, and identifiers. Lists can be constructed by using the function *SETQ* which takes two arguments, a list name, or identifier, and a value for the list, viz., a list of elements which can be any of the types above. The use of SETQ has the additional side effect of making the list name a *FLUID VARIABLE*. This simply means that it has dynamic or non-*LOCAL* scope and is more widely accessible, in contrast with *LOCAL VARIABLES* which are bound to fixed values inside the functional environment in which they occur. Since the default in CL is to treat variables as local, *FLUID VARIABLES* have to be declared, using the function *FLUID*, in order for the compiler to generate the correct code for dynamic access. However, we do not need to declare the actual size of these data structures or any other types of data objects that we want to construct since LISP automatically manages memory space by using a garbage collector to "sweep away" objects that are no longer in use.

The syntax for data representation in CL is to enclose the list in round brackets and separate the elements of the list with spaces. there is a simple isomorphism between list structure in CL and its representation in Machine Memory, a representation referred to as *box-diagram* notation which constructs lists as consecutive boxes called *CONS CELLS* terminated by a special identifier, which is NIL or the empty list. Each cell is represented by a box with the first half specifying one element of the list and the second half pointing to another box that represents the *REST* of the list. This notation is sometimes also called *DOTTED-PAIR* notation. To give an example, the list (1 2 3) can be represented in *box-diagram* notation as:



List forms are interpreted by a LISP function called *EVAL*, which predictably evaluates lists to their ASSOCIATed values. To take an example: (SETQ roots '(ktb drs qtl d;rb svkr)) is a

function call that initializes the list *roots* to a value which is the set of quoted elements between round brackets; (FLUID roots) declares that the list name *roots* is a FLUID VARIABLE with dynamic scope and thus will be accessible by all parts of the program; (EVAL 'roots) has the effect of LISTing all the elements of *roots* as the value ASSOCIATed with that name. It is also possible to create lists which are called *association lists*. An example of these is the list of dotted-pair cells: ((1 . 2) (3 . 4) (5 . 6)).

Another special example of data-structure representation is the *stack*, which is a kind of organized list structure. It is organized on a similar basis to self-service trays in cafés, viz., on the principle of first-in last-out: elements are added to an initially empty list as they are given, with the first element going to the bottom of the list, the next on top of it, and so on. Addition to this special list is achieved by using the function *PUSH*. The function *POP* can be used to take elements off the stack, one at a time on the basis of last-in first-out.

I.4 DATA ACCESS AND DATA MANIPULATION IN CL

CL provides fast *primitive*, or *built-in*, functions for direct access to data structures by user or by program. Each CONS cell in a list structure has two components called *CAR* and *CDR*. Particular positions in the list can be obtained using the function *CAR* which RETURNS the CAR component, or *head*, of a list, and the function *CDR* which RETURNS the CDR component of the list, or its *tail*. Any combination of these can be used to RETURN corresponding positions in the list. Thus, *CADR* would RETURN the first element of the REST of a list, *CDDR* the REST of the REST of a list, and so on. The function *LAST* can be used to obtain the last element of a list, and the function *LENGTH* to obtain the number of elements in it.

Other facilities are available for manipulating lists, either creatively or destructively. Creative functions, such as *LIST* simply, make a list out of one or more items or out of a list of lists. *APPEND* adds elements to a list, and *CONS* creates a new list out of specified input variables. *REVERSE* reverses a list on the basis of last becomes first, *DELETE* deletes an element from a list, and *SUBSTITUTE* substitutes a new element for an old one in the list. But these last functions do not alter the structure of their list arguments in the environment. Destructive functions such as *RPLACA* and *RPLACD* do irrevocably alter the structure of a list by replacing its CAR or CDR respectively with new elements.

There are also facilities for comparing lists in CL such as the functions *EQ* and *EQUAL* which verify equality between two lists but also between two ATOMS such as identifiers or numbers. The function *EQCAR* can be used to compare the first elements of a list to other objects. The function *PAIRP* checks if its argument is a list. The function *IDP* verifies if its argument is an identifier and the function *STRINGP* checks if its argument is a string. *UNION*

applied to two lists RETURNS the elements in the first list that are not in the second plus the second list. *SETDIFF* RETURNS the elements that are in the first but not in the second list and the function *XN* RETURNS the intersection, or common elements, of two lists.

Facilities also exist for verifying membership in a list via the use of *MEMBER* or *MEMQ*. Association lists can be accessed directly using the function *ATSOC*. Further facilities exist for handling ATOMS such as *QUOTE* which RETURNS the ATOM without its value, *EXPLODE* which converts an ATOM to a list of its constituent characters separated by spaces. *PACK* and *MKATOM* can be used to repair the damage, i.e., to reconvert or compress lists of such characters into ATOMS. Other functions such as *PRINT* simply print ATOMS or other objects as the arguments and the value. To give a few examples of all these functions using the list of roots defined above:

(CAR roots) =	ktb.
(CDR roots) =	(drs qtl d;rb svkr).
(CADR roots) =	drs.
(LAST roots) =	(svkr).
(LENGTH roots) =	5.
(LIST roots) =	((ktb drs qtl d;rb svkr)).
(REVERSE roots) =	(svkr d;rb qtl drs ktb).
(COPY roots) =	(ktb drs qtl d;rb svkr).
(DELETE 'drs roots) =	(ktb qtl d;rb svkr).
(APPEND '(kbr) roots) =	(kbr ktb drs qtl d;rb svkr).
(CONS 'drs 'ktb) =	(drs . ktb).
(SUBST 'kbr 'ktb roots) =	(kbr drs qtl d;rb svkr).
(EQUAL 'kbr 'ktb) =	NIL.
(EQCAR roots 'ktb) =	T.
(PAIRP roots) =	T.
(UNION '(kbr ktb drs) roots) =	(kbr ktb drs qtl d;rb svkr).
(SETDIFF '(kbr ktb drs) roots) =	(kbr).
(XN '(kbr ktb drs) roots) =	(ktb drs).
(MEMQ 'kbr roots) =	NIL.
(ATSOC 3 '((1 . 2) (3 . 4)) =	(3 . 4).
(EXPLODE 'kbr) =	(k b r).
(QUOTE 'kbr) =	kbr.
(PRINT 'kbr) =	kbr, %%% *print name.*
	kbr. %%% *value.*
(IDP 'kbr) =	T.
(STRINGP "abc") =	T.

(DE *COMPRESS* (x) (*CLEARBUFF*) (PACK x NIL) (MKATOM)).

(DE *MKWORD* (list) (MAPC list *COMPRESS*)).

(*MKWORD* '(k b r)) = kbr.

Other functions such as arithmetic ones are available to handle numerical ATOMS such as integers, rational numbers, and so on. Some functions operate on numbers. For instance *PLUS, DIFFERENCE, TIMES, QUOTIENT, REMAINDER,* and *DIVIDE,* can be used as would be expected. The functions *ADD1* and *SUB1* respectively add 1 and subtract 1 from their arguments. There is a large number of such arithmetic functions in the system. Other functions can be used to verify the status of, rather than operate on, a number. For instance:

(ZEROP 0) = T.
(ONEP 1) = T.
(NUMBERP 3) = T.
(EVENP 2) = T.
(MINUSP -4) = T.

Other functions exist to compare numbers as follows:

(GREATERP 4 3) = T.
(LESSP 3 4) = T.
(GEQ 4 4) = T.
(LEQ 3 4) = T.
(NEQ 2 2) = NIL.

I.5 PROPERTY ASSIGNMENT IN CL

Besides identifiers with dynamic-scope status, it is possible to create other types of interesting identifiers in LISP. These are ATOMS which carry properties and features. *ASSIGNing* properties to identifiers is a simple matter in CL. Using the function *PUT* which takes three arguments, we can initialize a *property list* for a specified identifier with a specified value as follows:

(PUT *property-label identifier property-value*).

The effect of this function call is to ASSIGN the property value with the identifier in the property list, also called a *PLIST*, which has the name *property-label.* When we have a large database or list of identifiers it may become tedious to ASSIGN properties individually to each identifier, so the function *MAPC* is provided as a kind of abbreviation to ASSIGN property values to a set of identifiers with a common feature. Thus, the calls:

- (DE *FUNCTION-NAME* (identifier)

(PUT property-label identifier property-value)),

- (MAPC list-of-identifiers *FUNCTION-NAME*),

have the effect of mapping the property value to each element in the list of identifiers with one single function call. The PLIST thus created is a global mutable data-structure facility which can also be used in CL for storing a large variety of data items in addition to the other data structures described above. We can also see that in CL, unlike in most other artificial languages, identifiers have real existence at run time with a print name, a value, and possibly a property list, and are not just a set of stack references with code keys and so on.

The property values in PLISTS are not irrevocable and can in fact be retrieved, MODIFIed or even destroyed at will. Access to PLISTS is obtained by the function *GET*, which takes two arguments as in: (GET property-label identifier) which RETURNS the current property value of the identifier. Verification of current property values for a given property label can be obtained at will using the call: *(PLIST property-label)* which RETURNS a list of all current property values for the property label. MODIFICation of property values can be achieved by resetting the property list using the function PUT. More destructive functions exist which allow the reinitialization of PLISTS or of property values. The function *SETPLIST* can be used to reset a PLIST to a NIL value as follows: *(SETPLIST property-label NIL)*. The function *REMPROP* removes individual property values for a specified identifier from its PLIST as in: (REMPROP identifier property-value). Taking the example of *roots* (defined above) we can define a function called $FUNCTION_1$ as follows:

(DE $FUNCTION_1$ (v) (PUT 'TRANSITIVITY v 'monotransitive)).

The statement (MAPC roots $FUNCTION_1$) has the effect of ASSIGNing each element or identifier in the list *roots*, the *TRANSITIVITY* value *monotransitive* in the property list with the label *TRANSITIVITY*. Now we can ask LISP: (GET 'TRANSITIVITY 'ktb) to obtain: *monotransitive*. We may later wish to *MODIFY* that value just for 'ktb', for instance, so we call PUT again as in: (PUT 'TRANSITIVITY 'ktb 'intransitive) which will erase the initial TRANSITIVITY value for 'ktb' and replace it with the new one. A call to REMPROP will simply remove the initial value altogether: (REMPROP 'TRANSITIVITY 'ktb).

II PROCESSING GENERALITIES: INITIAL CLOSED SUBROUTINES

Having described a repertoire of system-defined functions in Cambridge LISP which provides facilities for data representation, data access, and data manipulation, as well as facilities for property ASSIGNment, and function definition by the LISP user, we now need to make use of these facilities in order to build up some general but essential initial subroutines that are

necessary for the operation of our morphological processor.

Thus, in encoding the grammar rules and the data necessary for the MA of Arabic, we need first of all to define basic subroutines that can be used to handle frequent but essential primitive operations. These operations can then be invoked from wherever they are needed in the program. We will refer to these independently-defined operations as *closed* subroutines as opposed to *open* subroutines which are defined every time they are needed at a given point in a given program. The advantage of *closed* subroutines is that they are defined only once and allow a given a system to have a modular form that can be updated independently.

II.1 SEGMENTATION

One such necessary subroutine for morphological processing is *SEGMENTation*. Before we can automatically recognize a given *inputword*, which is at this point an unknown entity of unknown LENGTH and structure, we need to be able to recognize its constituent parts. Since an inputword is written as one single unit, we need an operation which can break it down into segments. The question is then how long is a given segment going to be? Since the LENGTH of such segments is unknown at this point in the proceedings, our SEGMENTation function can use a variable LENGTH argument that we will call an *index*.

Initially then, we can define a function called *SEGM* which takes two arguments, an inputword and an index or numeral indicator which is to be a positive whole number that is exactly the number of characters of the required segment. The main operation of SEGM is handled by a recursive function which we will call *RECURSE* and which takes three arguments, the index, the modified EXPLODed inputword, and a stack, or register. RECURSE picks up consecutive first elements off the front of the list of constituent characters of the inputword and deposits them in the stack until the LENGTH of the stack—which is EQUAL to the number of characters stored there—is EQUAL to the index, whereupon RECURSE hands back control to SEGM which ASSEMBLes the characters in the stack into a new word and RETURNS a dotted-pair list containing that new word as the required segment and the REST of the inputword. However, to prevent RECURSE from going into an infinite regress, SEGM needs to specify a control structure which specifies *exit* conditions to stop the application of RECURSE. There are in fact two patent conditions where RECURSE can be stopped: 1) the inputword list is empty; or 2) the index is zero or is EQUAL to or GREATER than the LENGTH of the inputword list. We can now outline an algorithm for SEGM and RECURSE.

As defined below, SEGM can only take characters off the front of the word. However, we might wish to be able to examine segments off the end of an inputword, i.e., proceeding from right-to-left. Using the same arguments, we can define parallel functions to SEGM and RECURSE to achieve this goal. We will call these *SEGMV* and *RECURSEV*. The algorithm

ARGUMENTS: an inputword, w and an index, i.

OBJECTIVE:

SEGMENT a word into a prefix of LENGTH i and the REST of the word.

LOCAL VARIABLES: L for LENGTH and reg as a stack.

BASIC METHOD:

STEP 1: SET the inputword, w to be the modified EXPLODed inputword.

STEP 2: SET L to be the LENGTH of the word, w.

STEP 3: IF w is empty or IF i is zero or EQUAL to or GREATER than L, then FAIL.

STEP 4: ELSE RECURSE on w using the index i and the stack reg.

RECURSE: %%% main recursion in SEGM.

STEP 4.1: IF the LENGTH of the stack is EQUAL to the index then

RETURN a dotted-pair list which is the result of ASSEMBLing the REVERSE of the stack and ASSEMBLing the REST of the inputword, w.

STEP 4.2: ELSE DO:

(a) ADD the 1st element of the inputword, w to the stack, and

(b) Apply RECURSE to the REST of the inputword, w using the index, i and the stack, reg.

EXAMPLE OF OUTPUT: (SEGM 2 'sa:akotubu) = (sa . :akotubu).

SCHEME 1: SEGM: A Recursive Scheme for Left-to-Right SEGMENTation

to implement SEGMV and RECURSEV will then be exactly the same as for SEGM and RECURSE except that in STEP 1 the inputword, w is SET to be the REVERSE of the modified EXPLODed inputword and in STEP 4.1 the PAIR RETURNed is the result of ASSEMBLing the stack and ASSEMBLing the REVERSE of the word.

Besides separate initial and final segments, we might need to examine both segments simultaneously. This can be achieved by combining the results of SEGM and SEGMV. Here, we will need an index i for the LENGTH of the initial segment or the prefix and an extra index j for the LENGTH of the final segment, or the suffix. We can then define a function *MNSEGM* for main segment, which takes two indexes and the inputword and RETURNS the result of applying SEGM to the inputword and SEGMV to the second element of the PAIR RETURNed by SEGM, i.e., the REST of the inputword. The definition of MNSEGM is given below.

ARGUMENTS: an inputword, w and two indexes, i & j.

OBJECTIVE:

SEGMENT a word into three segments: a prefix, a suffix, and the REST of the word.

LOCAL VARIABLES: L & m.

BASIC METHOD:

STEP 1: SET L to be the LENGTH of the inputword, w.

STEP 2: IF either of the indexes i or j is zero or the sum of i and j is EQUAL to or GREATER than L, then FAIL.

STEP 3: ELSE DO:

(a) SET m to be the result of applying SEGM to w using i, and

(b) RETURN a list which is m followed by the result of applying SEGMV to the 2nd element of m using j.

EXAMPLE OF OUTPUT:

(MNSEGM 1 4 'yadorusuwna) = ((y . adorusuwna) uwna . adorus).

SCHEME 2: MNSEGM: A Scheme for Combining Left-to-Right and Right-to-Left SEGMENTation

As defined here the SEGMENTation routine is a powerful recursive subroutine which is able to take an inputword of unknown LENGTH, L and one or two indexes, i and j and RETURN a prefix and the REST of the word, a suffix and the REST of the word or a combination of a prefix, a suffix, and the REST of the word. So that the SEGMENTation function is able to cope with all these values of i, j, and the REST of the word:

- For SEGM: $1 \leq i < L$ and $1 < \text{REST (word)} < \infty$.
- For SEGMV: $1 \leq i < L$ and $1 < \text{REST (word)} < \infty$.
- For MNSEGM: $1 \leq i < L$, $1 \leq j < L$, and $2 < \text{REST (word)} < \infty$.

II.2 AUXILIARY SUBROUTINES: TRANSCRIPTION AND WORD ASSEMBLY

Now, we need to define the auxiliary subroutines already mentioned: *MODIFIED-EXPLODE* and *ASSEMBLE*. We will call these *SCRIPT* and *MKWORD* respectively where *mk* is an abbreviation for make. The Arabic transcription system used here presents us with a problem as some special characters, namely these: { ; - / +} need to be considered as part of the consonants preceding them and not as independent characters. This has implications for calculating the

LENGTH of a word since we do not wish to include them in that calculation. We can define a procedural loop called *SCRIPT* which takes an Arabic inputword as its sole argument in order to cope with these special characters. The definition of *SCRIPT* is as follows:

ARGUMENTS: an Arabic inputword, w.

OBJECTIVE:

hand over an EXPLODed word to the function ARABIC and RETURN the result of applying ARABIC.

BASIC METHOD:

STEP 1: SET word, w to be the EXPLODed word.

STEP 2: Apply ARABIC to the list of the EXPLODed word.

ARABIC: %%% main recursion in SCRIPT.

ARGUMENTS: an EXPLODed inputword list, w.

OBJECTIVE:

SEPARATE ordinary single characters and group PAIRS of single characters followed by special characters.

LOCAL VARIABLE: ans.

BASIC METHOD:

STEP 1: IF the 2nd element of the list, w is one of the special Xers: {; - / +} then DO:

(a) CLEAR the internal machine buffer,

(b) ASSEMBLE the 1st and 2nd elements of the list as one Xer and make an ATOM of both,

(c) SET ans to be a PAIR of the result of (b) and ans, and

(d) SET the list to be the REST of the list.

STEP 2: ELSE DO:

(a) SET ans to be the result of ADDing the 1st element of the list to ans, and

(b) SET the list to be the REST of the list.

STEP 3: REPEAT 1 and 2 until the list is empty, then RETURN the REVERSE of ans without leaving a COPY of it behind.

EXAMPLE OF OUTPUT: (SCRIPT 't-alat-a) = (t- a l a t- a).

%%% LENGTH will be 6 not 8.

SCHEME 3: SCRIPT: An Iterative Scheme for Coping with the Arabic TRANSCRIPTION System

We can now define the ASSEMBLing function *MKWORD*. This is a function that takes

a list from a SCRIPTed word and reASSEMBLes it into a reconstructed word. An iterative definition for MKWORD can be outlined as follows.

ARGUMENTS: a list of the characters of a SCRIPTed inputword, w.

OBJECTIVE:

reASSEMBLE or reconstruct (by COMPRESSion, or DELETion of spaces) the characters into a word.

BASIC METHOD:

STEP 1: IF the list is empty, then FAIL.

STEP 2: ELSE DO:

(a) CLEAR the internal machine buffer,

(b) For each Xer in the list, ADD the Xer to the buffer, and

(c) RETURN the result of making an ATOM out of all the Xers in the buffer.

EXAMPLE OF OUTPUT: (MKWORD '(t- a l a t- a)) = t-alat-a.

SCHEME 4: MKWORD: An Iterative Scheme for ASSEMBLing a SCRIPTed word

We can now illustrate the logical overall operation of the SEGMENTation routine SEGM in Figure 1 below.

II.3 MATCHING

Another necessary subroutine for morphological processing is an operation which, given an inputword and a data structure such as a pattern, can test if there is a match between them. From the descriptions in Chapter 1, § II.3.5, we can in fact deduce a formal specification for patterns as in Table 1 which follows Figure 1.

The above specification of a pattern-constituent structure as a sequence of primary data structures: consonant tokens, c, and *lexical* (i.e., specified) vowels and semivowels, is one where we can view the tokens as slots and the lexical items as variables. We can then define a recursive *MATCHing* function that we will call *MATCHC* which, using that specification, takes two arguments: a pattern and an inputword. The basic idea is, going from left-to-right, to find if for each consonant slot in the pattern, there is in the same position in the inputword a consonantal realization, and if for each vowel or semivowel specified in the pattern, there is in the same position in the inputword an identical lexically specified element. The definition of *MATCHC* follows Table 1.

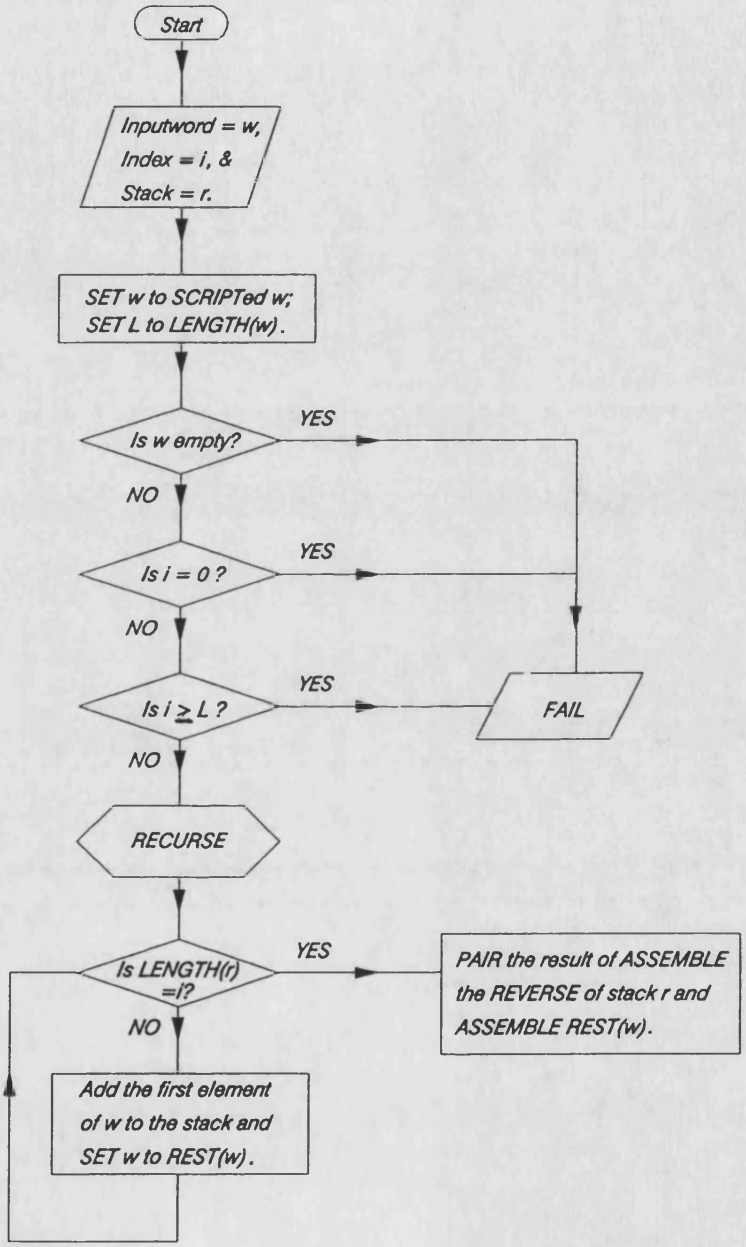


FIGURE 1: A Representation of the SEGMENTation Subroutine

DESCRIPTION TYPE	a shallow, or semi-abstract, structure.
STRUCTURE	*BASIC STRUCTURE:* a sequence of consonant tokens and lexical (i.e., specified) vowels and semivowels. *CONCATENATION ORDER:* as given, left-to-right. *PATTERN TYPES:* (a) *Sound:* with no semivowel contents. (b) *Defective:* with semivowel contents. *EXAMPLES:* (a) cacac. (b) acuwc.
CONSTRAINTS	*GRAPHOTACTIC CONDITIONS:* none. *LENGTH:* unknown but ≥ 1. *BOUNDARIES:* unknown.
ROLE	constrain structurally and lexically the realization of CFs and NFs in conjunction with Verbal and V-Roots and N-Stems.

TABLE 1: A Formal Specification for a Pattern

ARGUMENTS: an EXPLODed pattern and a SCRIPTed inputword.

OBJECTIVE:

test if there is a MATCH between the pattern and the inputword.

BASIC METHOD:

STEP 1: IF the inputword list is empty, then the list of pattern Xers must be empty.

STEP 2: IF the 1st Xer of the pattern is the token *c* (for consonant), then IF the 1st Xer of the inputword list is a consonant, then apply MATCHC between the REST of the pattern Xers and the REST of the inputword list.

STEP 3: ELSE, IF the 1st Xer of the pattern is identical with the 1st Xer of the inputword list then apply MATCHC between the REST of the pattern Xers and the REST of the inputword list.

STEP 4: ELSE FAIL.

%%% MATCHC is a powerful recursive MATCHing method but is too unrestricted, so we need another function MATCHX which, given two arguments: a pattern and an inputword, has to transform them into lists by EXPLODing and SCRIPTing them and to impose preliminary conditions to constrain the MATCHing routine before going ahead with the MATCH. The definition of MATCHX is as follows:

ARGUMENTS: a pattern and an inputword.

OBJECTIVE:

transform both pattern and inputword into lists to be MATCHed and impose preliminary conditions.

LOCAL VARIABLES: L and n.

BASIC METHOD:

STEP 1: SET the pattern to be the EXPLODed pattern.

STEP 2: SET the inputword to be the SCRIPTed inputword.

STEP 3: SET L to be the LENGTH of the pattern.

STEP 4: SET n to be the LENGTH of the inputword.

STEP 5: IF L is EQUAL to n then RETURN the result of applying MATCHC between the pattern and the inputword.

STEP 6: ELSE FAIL.

%%% MATCHX applies between one pattern and one word at a time. However, we have to search for a pattern that MATCHes the inputword in a list of patterns. So we need a function MATCHW which, given a list of patterns and an inputword, can search for a MATCH in that list.

ARGUMENTS: a pattern list and an inputword.

OBJECTIVE: search the list for a pattern that MATCHes the inputword.

BASIC METHOD:

STEP 1: IF the list is empty then FAIL.

STEP 2: ELSE apply MATCHX between the 1st element of the list and the inputword, then

(a) IF the MATCH is successful, then RETURN the 1st element of the list.

(b) ELSE apply MATCHW between the REST of the list and the inputword.

EXAMPLES OF OUTPUT:

(MATCHX 'cawwac 'kawwan) = T, (MATCHX 'cayyac 'kawwan) = NIL.

(MATCHW '(cacac cawwac ca|c) 'kawwan) = cawwac.

SCHEME 5: MATCHW: A Recursive Scheme for Left-to-Right MATCHing

The MATCHing subroutine is illustrated in Figure 2 below:

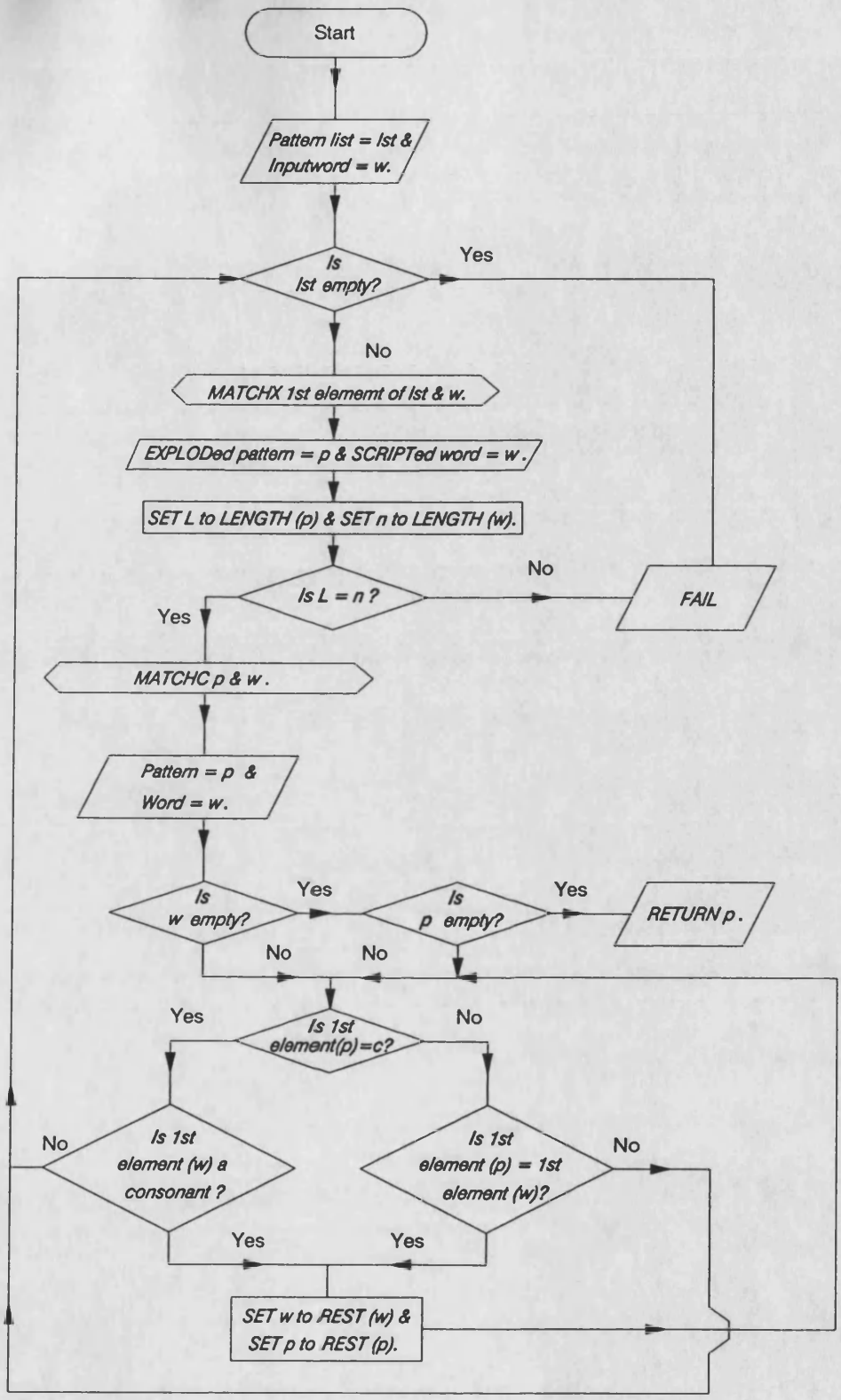


FIGURE 2: A Representation of the MATCHIng Subroutine

II.4 CHARACTER PICKING

During processing we might wish to single out one particular character in a given position such as first or second or penultimate. The aim of such a selection might vary from a simple comparison to operations such as SUBSTITUTion, DELETion or COPYing. We then need a subroutine which given an inputword can pinpoint the required character. We define a recursive function *NTHCHAR* which takes two arguments, an index which is the position required and the SCRIPTed word:

ARGUMENTS: a numerical index and a list of the SCRIPTed inputword.

OBJECTIVE:

RETURN the nth character, viz., that at the numerical front position (NTH) or back position (NTHV) in the inputword.

BASIC METHOD: %%% NTHCHAR.

STEP 1: IF the list is empty, then FAIL.

STEP 2: IF the index is EQUAL to 1, then RETURN the 1st element of the list.

STEP 3: ELSE SET the index to be the index MINUS 1 and apply NTHCHAR between the index and the REST of the list.

%%% It is too expensive to look too deep into the list while we can PICK characters either off the front or off the end of the word. So we define NTH for left-to-right and NTHV for right-to-left selection.

BASIC METHOD: %%% NTH.

STEP 1: SET the inputword to be the SCRIPTed inputword.

STEP 2: Apply NTHCHAR to the inputword using the index.

BASIC METHOD: %%% NTHV.

STEP 1: SET the inputword to be the SCRIPTed inputword.

STEP 2: SET the inputword to be the REVERSed inputword.

STEP 3: Apply NTHCHAR to the inputword using the index.

EXAMPLES OF OUTPUT: (NTH 2 'kutiba) = u, (NTHV 1 'kutiba) = a.

SCHEME 6: NTH and NTHV: Recursive Schemes for Left-to-Right and Right-to-Left Character PICKing

The character PICKing subroutine is illustrated in Figure 3 below:

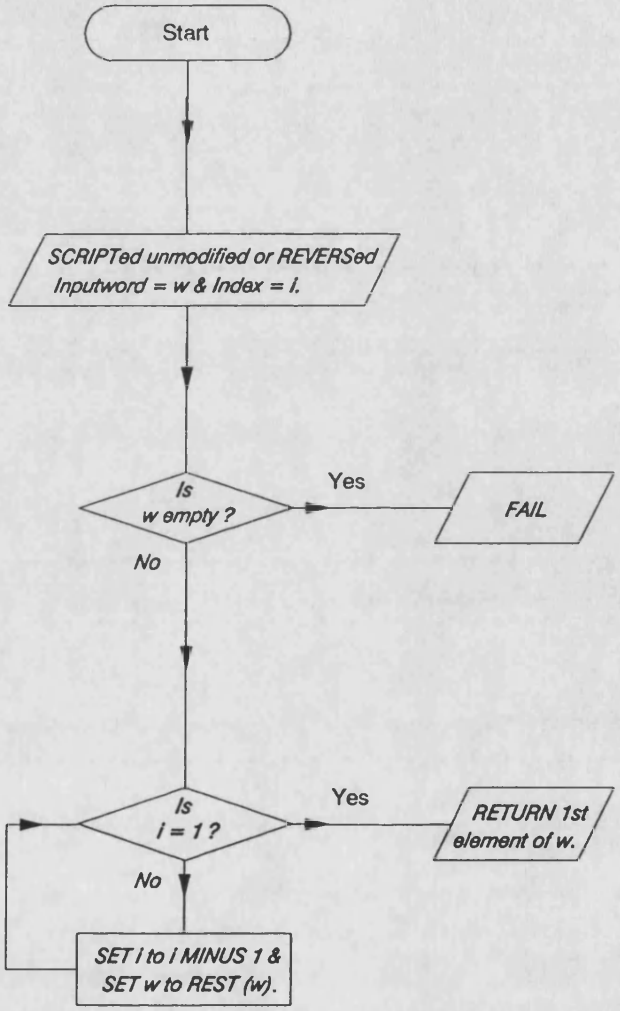


FIGURE 3: A Representation of the Character PICKing Subroutine

Besides SEGMENTation, MATCHing, character PICKing, word ASSEMBLY, and TRANSCRIPTION subroutines, we mentioned in passing operations such as *DELETion*, *SUBSTITUTion*, and *COPYing*. These are essentially system-defined functions: *DELETE, SUBSTITUTE*, and *COPY* which we modified slightly to build *SUBST1* which SUBSTITUTES characters in an inputword from the end, *SUBST2* which SUBSTITUTES characters from the front, *DELETEV* which ASSEMBLes an inputword after SCRIPTing it, REVERSing, and DELETing characters from its end and *DELETEW* which DELETes characters off the end of a list of characters rather than a word.

The functions *SETQ* for SETting variables, *PUT* for ASSIGNing properties, *ATSOC* for ASSOCIATing a variable with a list and the functions *EXPLODE, REVERSE, MEMBER, LENGTH,* and *RETURN* are again all system-defined functions. We also needed to define a recursive function *MAKESTRING* which is parallel to MKWORD but takes a string and one or more words as arguments instead of a list of characters. *MAKESTRING* RETURNS a new string that is the old one and the added word or words. Adding words to a list involves using the system-defined function CONS.

==0==<*****>==0==

Part II

THE VERB COMPLEX IN M.S.A.

Chapter 4

A FORMAL MODEL OF THE V-COMPLEX

INTRODUCTION

In this Chapter, we need to provide a linguistic précis of the structure of the Verb in M.S.A. In Section I, we seek an optimum typology of this Verb using these structural criteria: the syntactic requirements of the Verb, its *radical*, or *root*, types, its morphological and graphological radical and pattern structure. We will also allude to semantic criteria, such as time reference. All this will allow us to define the Arabic *Verb* system.

We will then delineate the scope of the Verb analysis and select a set of Verbs from the various types and classes described. A list of conjugations in the different TENSES, MOODS, and VOICES for these Verbs will be provided (in Vol. II, App. A, § A).

The conjugations will allow us to define more formally (in § II) the precise constituent structure of the Verb in M.S.A. By collapsing common forms together, we attempt to infer generalities and similarities between various types and classes of Verb. In these common forms, which we will call *Simple Conjugation Forms*, or *CFs*, we have to identify roots, patterns, and affixes with different distribution and allomorphic variation in the perfect and imperfect TENSES. For all such CFs we need to detail features and feature values for NUMBER, GENDER, PERSON, MOOD, VOICE, TRANSITIVITY, and CATEGORY.

However, will these generalizations be absolute? or will we have to refine them? How can we provide an adequate account of morphological HOMONYMY, a phenomena which we will see affecting the said features and feature values for roots, patterns, affixes, and CFs, as well

as the structural aspects of CFs.

We then need to examine the affixation and the conditions of affixation, if any, of affixes such as Accusative Pronouns and Future Particles to Simple CFs. This will allow us to extend the definition of CFs to cover Complex and other types of CF structure.

From these observations and descriptions, we should be able to deduce a body of formal structural rules which may be called *Morphological-Structure rules*; and to find ways of using these MS rules to generate valid M-Trees, and constraining their generative power to the generation of "all and only" valid M-Trees. These observations and rules could then be used to draw up a formal morphological model of the V-Complex and its logical structure, which the V-Processor could operate on to be able to parse such complexes optimally and efficiently.

I A LINGUISTIC DESCRIPTION OF THE VERB IN M.S.A.

I.1 A TYPOLOGY OF THE VERB IN M.S.A.

Traditionally, *Verbs* have been defined by their function in speech. For instance, in WINOGRAD, 1983: 52, we find a definition of the Verb as that part of speech which "signals the performance of an action, the occurrence of an event, or the presence of a condition". In the Arabic literature, we find mostly semantic definitions for the *'fiɛol'*, or "Verb". For instance, SHARIYF, 1979: 19, claims: "A Verb is that which has meaning with tense as an integral part of it.". Similarly, HASSAAN, 1979: 104, asserts: "A Verb is that which signifies an event and a tense.".

Verbs have also been defined by their morphological features. CRYSTAL, 1980: 374, defines the Verb as that "element which can display MORPHOLOGICAL contrasts of TENSE, ASPECT, VOICE, MOOD, PERSON, and NUMBER".

In M.S.A., the *Verb* involves a complex system of contrasts. Derivationally, it is a realization of an *abstract pattern* which carries information about TENSE, ASPECT, and VOICE. Inside the pattern is embedded a *root*, which carries information about the syntactic requirements of the Verb, or its TRANSITIVITY, and semantic content, or meaning. On the inflectional level, the Verb attaches with one or more affixes which indicate its NUMBER, GENDER, PERSON, and MOOD. We can use these syntactic and morphological criteria together with graphological descriptions to provide a detailed classification of Arabic Verbs.

I.1.1 Syntactic Requirements of the Verb: TRANSITIVITY

Arabic Verbs show a general two-way distinction between *transitive* and *intransitive*. Inside these two categories, further distinctions are possible based on the syntactic nature of the Objects or Complements required. Two broad distinctions of Object and Complement are possible: *Nominal* and *Prepositional* (both defined in Ch. 6). Further, membership in one or the other of the classes is a dynamic rather than a fixed one for each Verb. By modification to the vocalic structure and/or the pattern structure of a Verb, it is possible to *PROMOTE* or *DEMOTE* it to a different TRANSITIVITY rank. For instance, the Verb 'kataba', "he wrote", is *monotransitive*, 'kattaba', "he caused to write", is PROMOTed by causativization to *ditransitive* and 'kutiba', "it was written", is DEMOTed by passivization to *intransitive*.

There are also curious examples such as *ptransitive1* Verbs which require one Prepositional Object. For instance, the Verb 'd-ahaba', "he went", can be conjugated with all PERSONS in the active VOICE. However, under passivization, the Verb 'd-uhiba', "was gone", is restricted to impersonal use: it can be conjugated only with the third-PERSON-singular masculine; it requires no Objects, but has a Prepositional '*na/:ib fa/ɛil*', "Pseudo-Subject". In this respect, it is similar to the Latin Verb 'ire', "go", which has the active forms: '1 eo; 2 is; 3 it; 4 imus; 5 itis; 6 eunt'; but only the one passive form: '3 itur'.

Table 2 shows a detailed classification of the Verb in M.S.A. using the syntactic requirement criterion.

I.1.2 Root Types and Morphological Pattern Structure

All Arabic Verbs have radical segments called a *root*. We have already defined what a root is in Chapter 1, § II.3.4. We can say, moreover, that a *root* here should not be taken to mean "that part of the word left when all affixes are removed", CRYSTAL, 1980: 308, as in English. Rather an Arabic root is the set of non-vocalic segments which occur at the edges of syllables in the Verb. Normally, the number of these segments called *radicals* is three. However, it is possible to derive Verbs with more than three radicals by adding extra segments, a process referred to as *augmentation*. Thus, we can have two kinds of roots:

a. *Basic* roots which are indivisible into further parts. Basic roots are usually *Triliteral* with three radicals but may sometimes be *Quadriliteral* with four.

b. *Augmented* roots which are divisible into Basic roots and their augmentation. Each particular root in Arabic has one or more *patterns* associated with it and the augmentation of the root depends on those patterns. This involves:

TRANSITIVITY		SYNTACTIC REQUIREMENT	
		DEFINITION	EXAMPLE
INTRANSITIVE	*Intransitive*	requires no Objects.	'jalasa' "he sat"
	Ptransitive	impersonal use only; requires no Objects; but has a Prepositional Pseudo-Subject.	'd-uhiba' "was gone"
TRANSITIVE	*Monotransitive*	requires one Nominal Object.	'kataba' "he wrote"
	Ditransitive	requires two Nominal Objects.	'manah-a' "he granted"
	Xtransitive	requires one Nominal Object; and one Nominal or Prepositional Complement.	'h-asiba' "he thought"
	Cotransitive	requires one Nominal or Prepositional Complement.	'ka‖na' "he was"
	Nctransitive	requires one optional Nominal or Prepositional Complement.	'h-usiba' "he was thought"
	Ptransitive1	requires one Prepositional Object.	'nazala' "he arrived"
	Ptransitive2	requires one Nominal and one Prepositional Objects.	'manaʕa' "he prevented"

TABLE 2: Classification of the Verb in M.S.A. (1)

i. The reduplication of particular radicals at particular positions, e.g., the pattern 'faʕʕala-yufaʕʕilu' duplicates (by infixation) the middle radical of 'darasa', "he studied", to generate 'darrasa', "he taught".

ii. The supplementation of the radicals with specific segments among the set: {s, :, l, t, w, n, y}, e.g., the pattern ':isotafoʕala' adds the prefix ':isota', "to seek", to 'kataba', "he wrote", to generate ':isotakotaba', "he sought correspondence with".

Further, Augmented Verbs can be subdivided into *Mono-Augmented*, *Bi-Augmented*, and *Tri-Augmented*, i.e., Verbs that are augmented, respectively, by one, two, and three *syllemes*. We have coined the term *sylleme* which we define as a graphological unit that can be a sequence of a consonant and a vowel, the second consonant in a doubled sequence, or the character <|>. In Arabic, a sylleme is traditionally referred to as a *'h-arof'*, "letter". However, this term is ambiguous since it is used to denote a single character or letter and to denote what we have called a *sylleme*.

Table 3 below gives a full listing of the root types described above and the possible patterns that can be associated with them. The patterns ':ifoɛawoɛala', and ':ifoɛa|lla'—which is not listed here—are very rare in M.S.A. and more common in C.A. Note that WEHR, 1980: xiii, gives a different designation used by Western Orientalists for these patterns. NIϵMA, 1973: Vol. II, 219, gives yet another classification for them. Another issue that is not covered here is the semantic uses of each pattern. We refer the reader to the discussion of this topic in BAKIYR et al., 1975: 42–58.

I.1.3 Graphological Radical Structure

Roots are not only distinguished on the basis of whether they are Basic or Augmented, but also in terms of their graphological structure, i.e., the graphemic nature of the constituent radicals of the root. The constituent segments of the root can be wholly consonantal, in which case the root is said to be *Sound*, e.g., 'ktb', "write", or partly consonantal and partly semivocalic, in which case the root is said to be *Defective*.

Table 4 gives a brief classification of Verb roots in M.S.A. on the basis of the graphological nature of their radicals. Other types of Defective roots, which are not listed below, include *Sequential Double Defectives*—which have two consecutive semivowels, e.g.,'kwy', "to burn"—and *Discontinuous Double Defectives*, which have two non-consecutive semivowels, e.g., 'wsvy', "to inform".

I.1.4 Graphological Pattern Structure

As we hinted before, a *V-Pattern* is an abstract form usually listed in a third-PERSON-masculine singular perfect/imperfect form of the type 'faɛala- yafoɛulu' to represent Verbs such as 'kataba-yakotubu', "he wrote-he writes". In such forms, the convention is for <f> to represent the initial radical, < ɛ > the second radical, <l> the final radical and for all three letters to stand for any consonant or semivowel of the language. The rest of the pattern consists of:

a. Verb pronominal affixes at either end of the pattern which carry NGP and MOOD values.

ROOT TYPE			MORPHOLOGICAL PATTERN STRUCTURE	
			PERFECT PATTERN TYPE	EXAMPLE
BASIC	*Triliteral* I		faɛala	'qatala' "he killed"
			faɛila	'ɛalima' "he knew"
			faɛula	'karuma' "he was kind"
	Quadriliteral II		faɛolala	'baɛot-ara' "he scattered"
AUGMENTED	*Triliteral*	III	:afoɛala	':akorama' "he honoured"
		Mono-Augmented IV	fa|ɛala	'qa|tala' "he fought"
		V	faɛɛala	'kassara' "he shattered"
		VI	:inofaɛala	':inot;alaqa' "he departed"
		Bi-Augmented VII	:ifotaɛala	':ijotamaɛa' "he met"
		VIII	:ifoɛalla	':ih-omarra' "he reddened"
		IX	tafa|ɛala	'taqa|bala' "he faced"
		Tri-Augmented X	:isotafoɛala	':isotaqotala' "he sought to die"
		XI	:ifoɛawoɛala	':igoraworaqa' "he became tearful"
	Quadriliteral	*Mono-Augmented* XII	tafaɛɛala	'takassara' "it broke"
		Bi-Augmented XIII	:ifoɛanolala	':iforanoqaɛa' "it exploded"
		XIV	:ifoɛalalla	':iqosvaɛarra' "he shuddered"

TABLE 3: Classification of the Verb in M.S.A. (2)

GRAPHOLOGICAL RADICAL STRUCTURE			
TYPE		DEFINITION	EXAMPLE
S O U N D	*Solid*	All radicals are consonants.	'd;rb' "to hit"
	'Mahomuwz'	One of the radicals is the 'hamoza', <:>.	's:l' "to ask"
	Doubled Triliteral	The last two radicals are identical, or doubled.	'd/nn' "think"
D E F E C T I V E	*Quasi-Sound*	The initial radical is a semivowel.	'wqf' "to stop"
	Hollow	The middle radical is a semivowel.	'k|n' "to be"
	Weak	The final radical is a semivowel.	'bny' "to build"

TABLE 4: Classification of the Verb in M.S.A. (3)

b. Instances of vowels within it which indicate TENSE and VOICE contrasts. For example, the pattern 'yafoɛalu' indicates the values 3 (M) S, ind, prf, act. Other pattern types, e.g., ':ifoɛalo' indicate the imperative. Such patterns serve to derive and conjugate Verbs with *Subject Pronouns* and Nouns. Most Arabic Verbs are fully conjugable with these Pronouns and attach with pronominal affixes. These are referred to as *Plastic* Verbs. Certain Verbs however are *Aplastic* in that they have a frozen form with all *Subject Pronouns*. Plastic Verbs fall into two categories:

i. So. and Df.Vs that have *regular* conjugation in that the pattern has a constant form no matter which pronominal affix it is attached with. For instance 'ktb', "to write", has the perfect active pattern 'faɛal' with all Pronouns. Note that imperfect forms can have either present or future aspect. However the morphological form stays the same. The distinction is made by context or special morphemes, e.g., 'li', "in order to", 'sa', "going to", and 'sawofa', "will, shall". We return to this topic below.

ii. So. and Df.Vs, typically Dd., Ho., and Wk.Vs that undergo vocalic changes depending on which of the affixes they are attached with. For example, 'k|n', "to be", has the pattern 'fuɛ' with first- and second-PERSON Pronouns and the pattern 'fa|ɛ' with most third-PERSON Pronouns.

Table 5 shows a formal classification of the Verb in M.S.A. using all these structural criteria while Table 6 gives a listing of the 13 *Subject Personal Pronouns* in Arabic and their NGP values.

CONJUGATION	VOICE	TENSE	MOOD	GRAPHOLOGICAL PATTERN STRUCTURE	
				PATTERN TYPE AND VOCALIC CHANGES	EXAMPLE
PLASTIC	ACTIVE	*Perfect*		faɛala. given patterning.	'sakana' "he dwelled"
		Imperfect	*Indicative*	yafoɛulu. final vowel →<u>, etc.	'yasokunu' "he dwells"
			Subjunctive	yafoɛala. final vowel →<a>, etc.	'yasokuna' "he dwells"
			Jussive	yafoɛalo. final vowel →<o>, etc.	'yasokuno' "he dwells"
	PASSIVE	*Perfect*		penultimate vowel →<i>, all preceding vowels →<u>.	'svuriba' "it was drunk"
		Imperfect	*Indicative*	initial vowel →<u>, penultimate vowel →<a>.	'yusvokaru' "he is thanked"
			Subjunctive		'yusvokara' "he is thanked"
			Jussive		'yusvokaro' "he is thanked"
		Imperative		:ifoɛalo, :ufoɛulo. final-vowel changes.	':ih-omilo' "carry"
APLASTIC		*Perfect*		frozen or invariable form.	'ɛasay' "it may be"
		Imperative			'habo' "suppose"

TABLE 5: Classification of the Verb in M.S.A. (4)

PRONOUN		NGP VALUES		
		NUMBER	GENDER	PERSON
1	':ana|'	singular	masculine	first
	"I"		feminine	
2	':anota'		masculine	second
	"you"			
3	':anoti'		feminine	
	"you"			
4	'huwa'		masculine	third
	"he"			
5	'hiya'		feminine	
	"she"			
6	'nah-onu'	dual	masculine	first
	"we"		feminine	
7	':anotuma|'		masculine	second
	"you"		feminine	
8	'huma|'		masculine	third
8	"they"		feminine	
6	'nah-onu'	plural	masculine	first
	"we"		feminine	
9	':anotumo'		masculine	second
	"you"			
10	':anotunna'		feminine	
	"you"			
11	'humo'		masculine	third
	"they"			
12	'hunna'		feminine	
	"they"			

TABLE 6: Subject Personal Pronouns in M.S.A.

Table 6 shows 13 Subject Pronouns although there are actually only 12 different forms. This is because, the above listing is related to Verb CFs, which are in fact thirteen for each Verb. Thus, there are two different forms of the Verb for each GENDER value of number 8 above. In this sense, we can say that number 8 is homonymic. However 1, 6, and 7, which are also homonymic, have only one form of the Verb for both GENDERS. Further, number 6—which is normally one single form—is laid out as two for the purposes of this analysis, once for the dual, and once for the plural.

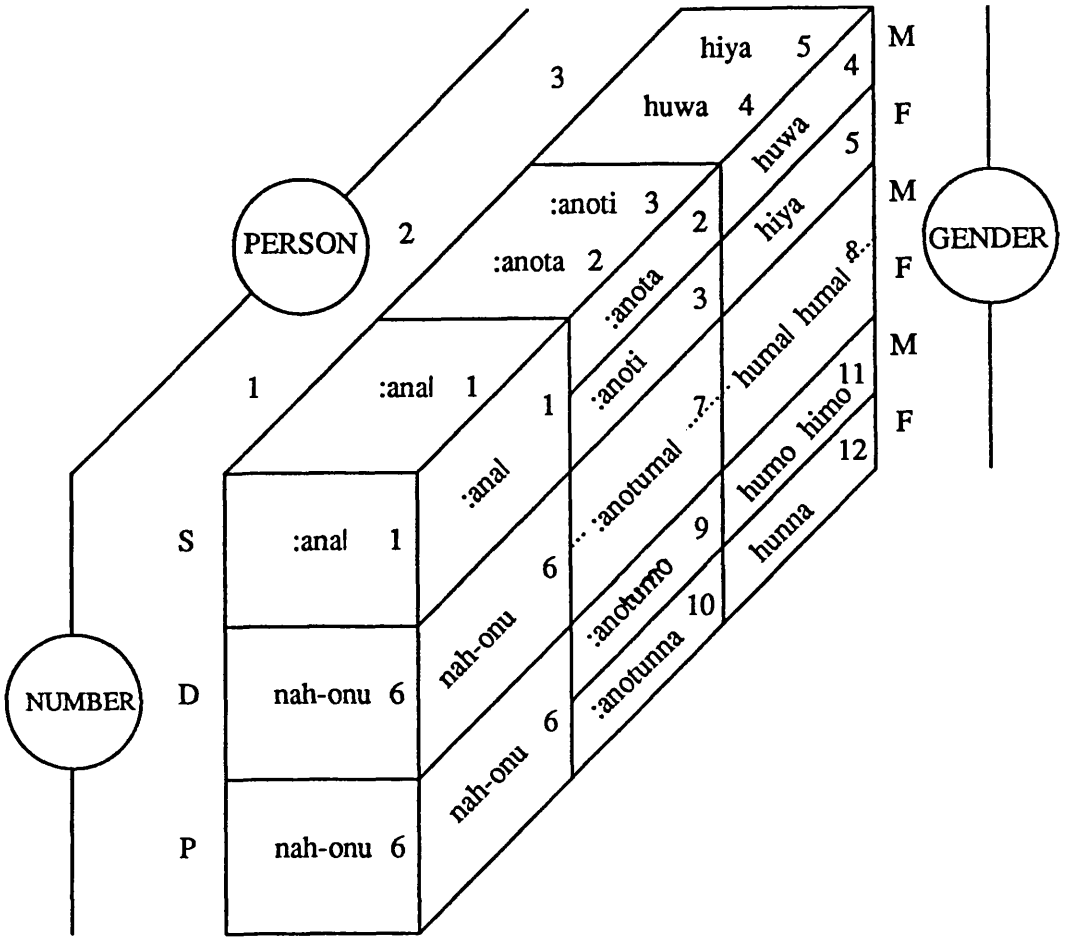


TABLE 7: Feature Dimensions for SPs in M.S.A.: Overlap

Table 7 above is adapted from the diagram given for English by WINOGRAD, 1983: 290, Figure 6–3, and shows more clearly the overlapping in NUMBER, GENDER, and PERSON for these Pronouns.

I.1.5 Other Considerations

Besides syntactic, morphological, and graphological considerations there are other criteria which, by definition, will not enter into our analysis, but which should be mentioned here for the sake of completeness.

a. The modality criterion can be used to divide Verbs into full Verbs and Auxiliaries. Auxiliaries can often be used in complex Verb phrases of the type 'ka|na kataba', "he had written". Strictly speaking, the analysis of such phrases, in fact, lies outside the scope of morphological study. It involves more than one morpheme and should be done at the syntactic level.

b. Semantically Arabic Verbs can be divided into many categories which include *Inchoative, Transmutative*, and *Factitive Verbs*, as well as *Verbs of Probability, Doubt, Certainty, Approximation, Praise, Vituperation, Surprise*, or *Wonder*, and *Verbs of the Heart*. These divisions refer to the semantic content of the Verb and its relations with its Complements. In this respect, ELTIKAINA, 1982: Ch. IV, lists up to twenty three semantic classes of Arabic Verbs.

c. There is currently great interest in temporal considerations—which involve the investigation of pragmatic-time reference—in the study of the Verb. For instance the whole of the 1988 Volume 14, Number 2, of *Computational Linguistics* was devoted to issues such as temporal reference and structure, semantics of TENSE and aspect, TENSE and discourse anaphor and quantification. Such topics do not lie within the scope of our present purely morphological investigations.

I.1.6 The Arabic Verb System

To summarize this introductory section on Verbs, we can say that the Arabic Verb is a complex system of overlapping levels of analysis. The derivational and inflectional levels enter into morphological and graphological interplay to generate surface realizations of underlying root and pattern forms that carry information about TRANSITIVITY, VOICE, TENSE, NUMBER, GENDER, PERSON, and MOOD. In addition, each form has radical type, pattern structure, regularity, and conjugation features. The overall system makes up the Verb CATEGORY. For instance, 'yafohamu', "he understands", indicates 3 (M) S, ind, imp, act, mtr, VB. It has a Basic Triliteral Sound root 'fhm', with regular conjugation, and an abstract pattern 'yafoɛalu'.

In addition, the root 'fhm' carries the semantic content "to understand" and the whole structure indicates the pragmatic-time reference "now". Table 8 below shows a multidimensional representation of the complexity of the Arabic Verb system.

I.2 SCOPE OF THE V-ANALYSIS

The present study has selected for analysis a set of Arabic Verbs that is representative of most of the classes of Verb described so far. The overriding concern has been selection and representativity rather than exhaustiveness, in the sense of including a large number of lexical items. Such exhaustiveness was restricted by time, space, and manpower constraints. Hence the total number of V-Roots was limited to 148, with 82 Basic and 66 Augmented roots. A complete list of these together with their classes can be found in Appendix A, § A.

The roots cover representative samples from all the Basic Triliteral, Sound (Doubled, Solid, and 'Mahomuwz') and Defective (Quasi-Sound, Hollow, and Weak) categories. They further exhibit all the different TRANSITIVITY values listed in Table 2. Also covered are Mono-Augmented Triliterals which we have divided into two groups:

a. *Causative*, which introduce the meaning of "cause to" and PROMOTE the TRANSITIVITY rank of the Verb to a higher rank, e.g., intransitive $\rightarrow\!\!\rightarrow$ monotransitive, monotransitive $\rightarrow\!\!\rightarrow$ ditransitive, and so on. For instance, 'fhm' is "to understand", 'fhhm' is "to cause to understand".

b. *Intensive*, which intensify the meaning of the Verb but preserve its TRANSITIVITY. For instance, 'ksr' is "to break", 'kssr' is "to shatter".

In Appendix A, § A, Basic Classes (in *a.* (1–2) and *c.* (3 and 5–10)) and Augmented classes (in *d.* (4)) have irregular conjugation while Basic classes (in *a.* (3–8), *c.* (1, 2), and *c.* (4)) and the rest of the Augmented classes have regular conjugation.

However, we have not covered the rare Quadriliteral roots and *Poly-Augmented* (Bi- and Tri-Augmented) Triliteral and Quadriliteral roots. Neither have we covered the imperative form of Verbs, temporal, modal or semantic distinctions. All these, although of an interesting nature, are not directly relevant to this study, which is more orientated towards the structural parsing of forms rather than their semantic analysis. Nevertheless, we believe that the methods used for the sample roots selected can be generalized to those that were not covered for lack of space.

Appendix A, § A, provides a list of CFs for one example in each category conjugated with all the Subject Pronouns, in the active and passive VOICE, in the perfect and imperfect TENSE, in the indicative, subjunctive, and jussive MOOD. Numbers there refer to Pronoun positions in Table 5.

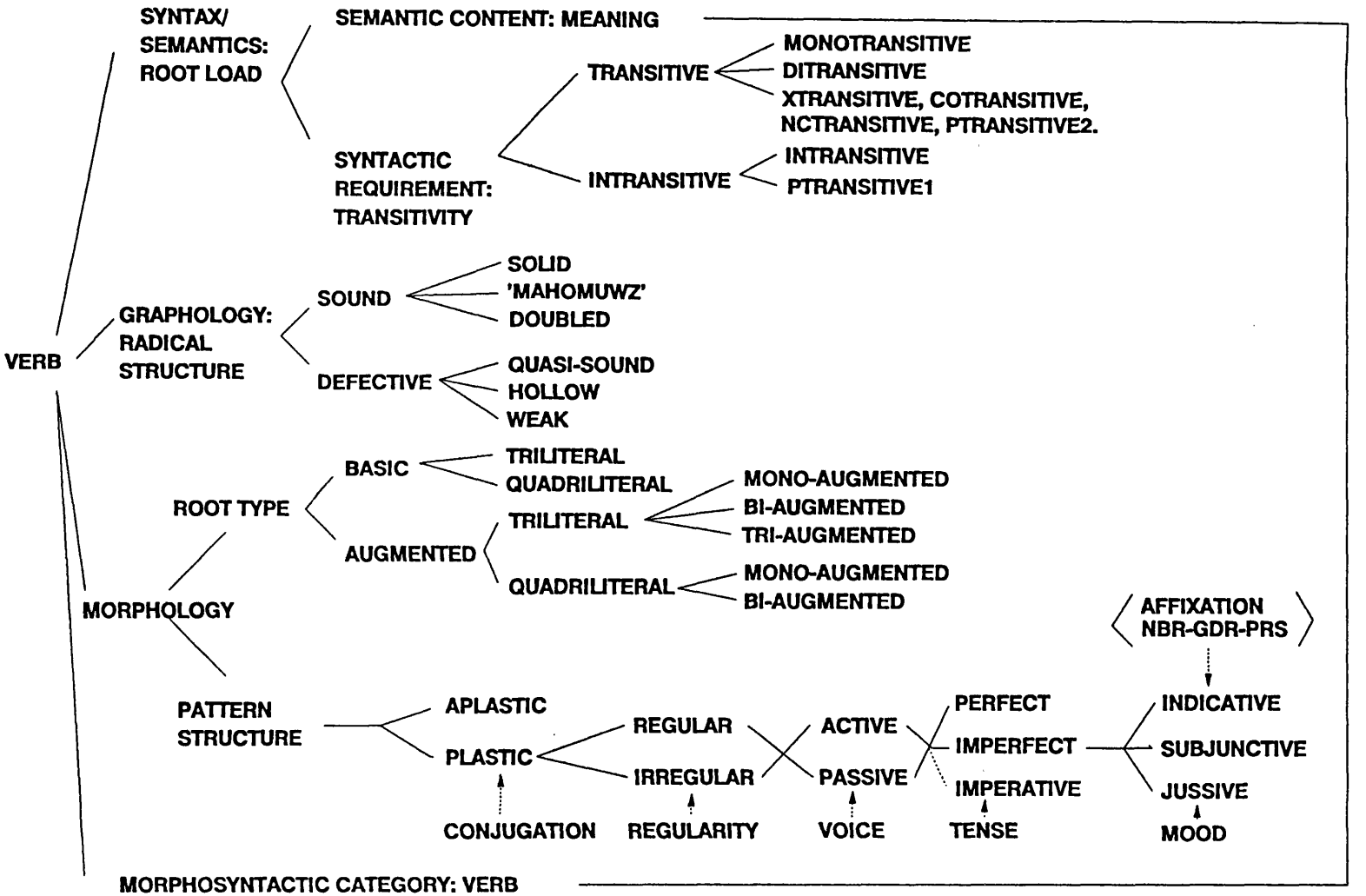


TABLE 8: The Arabic Verb System

II A FORMAL ACCOUNT OF THE VERB IN M.S.A.

II.1 GENERALIZATION

The automatic processing of Arabic Verbs, whether for generation or recognition, can be approached in several ways. We are faced at the outset with a choice between two approaches to the problem of data handling and representation. A dictionary approach to the representation of the above data would merely list all the different morphemes and allomorphs in a database. A purely programming method would have a minimum of dictionary entries and would rely instead on rules and procedures. It may in fact also be possible to conflate the two approaches. If we were to adopt the dictionary approach and list the various CFs for each root with each Pronoun in all VOICES, TENSES, and MOODS we would clearly end up with a table which was very cumbersome and expensive in terms of storage and manpower. Each root has at least 13 active perfect forms and another 13 passive perfect ones if applicable. It also has:

13 forms × 3 MOODS in the active imperfect and

13 forms × 3 MOODS in the passive imperfect if applicable.

This gives us a maximum of 104 different forms per Verb which means a maximum figure of 15,392 CFs for the 148 V-Roots selected here, which would mean for a lexicon of a mere 3,000 entries a conjugation table with 312,000 forms. Clearly this is undesirable and we need a more economic and efficient generalization of the data. In order to find such a generalization we need to carry out four main tasks:

a. Identify the relevant V-Affixes and their role in CF structure.

b. Identify the boundaries of the V-Patterns.

c. Identify the root forms within CFs.

Tasks *a.* to *c.* have to be carried out in such a way as to obviate the need for listing all possible forms; rather, we need to arrive at canonical forms that allow the construction of an optimal and cost-effective database.

d. Find a simple way to ASSOCIATE particular patterns with their appropriate root, or roots, in the database.

Addressing ourselves to tasks *a.* and *b.* involves the identification within CFs of sections that are common across TENSES, VOICES, and MOODS for all Pronouns. This means in effect treating CFs not as a single block but SEGMENTing them into relevant and sufficiently general sections. Taking the TENSE feature into consideration the perfect and imperfect forms are different enough for them not to form a basis for generalization. The former are divisible into

two parts, as they contain pronominal suffixes only, whereas the latter are divisible into three parts, as they have both pronominal prefixes and pronominal suffixes. The VOICE criterion does not allow a common active/passive form either, since these differ for each Verb in vocalic specification.

However, if we were to use the MOOD criterion, then we would notice that only the imperfect suffixes are variable for all the regular Verbs, whereas the rest of the CF is constant. Thus, we could account for the allomorphic variations of the suffixes separately, as opposed to the traditional method which includes them in pattern representations such as 'yafoɛalu' which indicate 3 (M) S, ind. Yet, the separation of affixes from the centre of the CF allows us to collapse the 13 different forms for each Verb in either TENSE or VOICE into one single form which can then be entered only once in the database instead of 13 times.

All in all, we need only four forms for a lexical entry: perfect active, perfect passive, imperfect active, and imperfect passive, instead of 104 forms for each Verb. This figure has to be slightly modified for irregular Verbs, which typically have two different forms for each conjugation, one form for first- and second-PERSON Pronouns and another for most third-PERSON Pronouns.

Affix stripping can then allow the discovery of the contextual conditions needed for MOOD specification and for NGP ASSIGNment. Hence, before identifying the pattern entries, we need to identify the exact forms that Verb affixes take in the different TENSES and MOODS.

II.1.1 Identification and Distribution of the V-Affixes

II.1.1.1 Suffixes of the Perfect Form

These can be divided into two groups, a constant group: {ato, a|, ata|} and a variable group: {{otu, tu}, {ona|, na|}, {ota, ta}, {oti, ti}, {otuma|, tuma|}, {otumo, tumo}, {otunna, tunna}, {ona, na}} where the deletion of <o> depends on the graphological context. The form 'uw|' can become 'awo|' and the forms {ay, a|} can indicate 3 (M) S instead of the regular suffix <a> with certain Df.Vs. We shall specify the exact conditions for these variations below.

Table 9 shows the configurations for the perfect suffix forms and their NGP values. Two kinds of allomorph are listed: default and contextual.

II.1.1.2 Affixes of the Imperfect Form

These affixes consist of four prefixes and various suffixes. The four prefixes: {:, t, y, n} are constant. The suffixes have the constant morphemes: {a|ni, a|} and the variable forms:

	SUFFIX FORM		
	DEFAULT	CONTEXTUAL	NGP VALUES
	otu	tu	1 (M, F) S.
V	ona|	na|	1 (M, F) D, P.
A	ota	ta	2 (M) S.
R	oti	ti	2 (F) S.
I	otuma|	tuma|	2 (M, F) D.
A	otumo	tumo	2 (M) P.
B	otunna	tunna	2 (F) P.
L	ona	na	3 (F) P.
E	uw|	awo|	3 (M) P.
	a	ay, a|	3 (M) S.
C			
O	ato		3 (F) S.
N			
S	a|		3 (M) D.
T			
A	ata|		3 (F) D.
N			
T			

TABLE 9: Perfect Suffix Form Configurations

{iyna, ayona, iy, ayo, uwna, awona, uw|, awo|, ona, na}. The regular suffixes {u, a, o} can become {uw, iy, ay} for certain Df.Vs. The variation depends on the graphological context. Table 10 shows the configurations for the imperfect affix forms and their NGP and MOOD values. For the suffixes, default forms are shown on the left of the column and contextual ones on the right.

II.1.2 Identification and Distribution of the V-Patterns

Having isolated the affixes attached to patterns within CFs, we now turn to the problem of variation within the pattern realization itself. We have previously (Ch. 1, § II.3.5) argued for the simplification of the conventional representation of patterns by removing the affixes from the representation, thus redefining the boundaries of the pattern, and by unifying the symbols: {f, ε, l} to <c>. This gives us the new simpler representation format: 'cacac-acocac' instead of the conventional 'faɛala-yafoɛalu'. Having redefined the pattern structure and its formal representation we can now identify the following patterns for the CFs listed in Appendix A, § A.

PREFIX FORM	SUFFIX FORM				NGP VALUE
	INDICATIVE		SUBJUNCTIVE	JUSSIVE	
:	u	uw	a	o	1 (M, F) S.
		iy		u	
		ay		i	
				ay	
n	u	uw	a	o	1 (M, F) D, P.
		iy		u	
		ay		i	
				ay	
t	u	uw	a	o	2 (M) S.
		iy		u	3 (F) S.
		ay		i	
				ay	
	iyna	ayona	iy	ayo	2 (F) S.
	a|ni		a|		2 (M, F) D.
					3 (F) D.
	uwna	awona	uw|	awo|	2 (M) P.
	ona na				2 (F) P.
y	u	uw	a	o	3 (M) S.
		iy		u	
		ay		i	
				ay	
	a|ni		a|		3 (M) D.
	uwna	awona	uw|	awo|	3 (M) P.
	ona na				3 (F) P.

TABLE 10: Imperfect Affix Form Configurations

II.1.2.1 The Perfect Pattern Forms

Table 11 below shows the distribution of perfect pattern forms for the different Subject Pronouns.

	REFERENCE in App. A, § A.	PRACT	PRPAS	NGP CONTEXT
1	a.(1, 2).	cacac	cucic	1, 2 (M, F) S, D, P. 3 (F) P.
2		cacc	cucc	3 (M, F) S, D. 3 (M) P.
3	a.(3, 8).	cacic	cucic	@.
4	a.(4).	cacuc	N/A.	@.
5	a.(5, 6, 7).	cacac	cucic	@.
6	b.(1).	caccac	cuccic	@.
7	c.(1).	wacac	wucic	@.
8	c.(2).	wacuc	N/A.	@.
9	c.(3).	cac	N/A.	1, 2 (M, F) S, D, P. 3 (F) P.
10		cayoc		3 (M, F) S, D. 3 (M) P.
11	c.(4).	cawic	N/A.	@.
12	c.(5).	cuc	N/A.	1, 2 (M, F) S, D, P. 3 (F) P.
13		ca|c		3 (M, F) S, D. 3 (M) P.
14	c.(6).	cic	N/A.	1, 2 (M, F) S, D, P. 3 (F) P.
15		ca|c		3 (M, F) S, D. 3 (M) P.
16	c.(7, 8).	cacay		1, 2 (M, F) S, D, P. 3 (M) D. 3 (F) P.
17		cac		3 (M, F) S. 3 (F) D. 3 (M) P.
18			cuciy	@ except ↓
19			cuc	3 (M) P.
20	c.(9).	caciy	N/A.	@ except ↓
21		cac		3 (M) P.
22	c.(10).	cacaw	N/A.	1, 2 (M, F) S, D, P. 3 (M) D. 3 (F) P.
23		cac		3 (M, F) S. 3 (F) D. 3 (M) P.
24			cuciy	@ except ↓
25			cuc	3 (M) P.
26	d.(1).	waccac	wuccic	@.
27	d.(2).	cayyac	cuyyic	@.
28	d.(3).	cawwac	cuwwic	@.
29	d.(4).	caccay		1, 2 (M, F) S, D, P. 3 (M) D. 3 (F) P.
30		cacc		3 (M, F) S. 3 (F) D. 3 (M) P.
31			cucciy	@ except ↓
32			cucc	3 (M) P.

TABLE 11: Perfect Pattern Forms

OBSERVATIONS:

We notice that the forms in Table 11 can be divided into two main groups:

1) A regular group including the So.Vs 3, 4, 5, & 6 above and the Df.Vs 7, 8, 11, 26, 27, & 28. In this group, only one pattern form is used for all SPs in both the active and the passive VOICE.

2) An irregular group that has one pattern form for certain SPs and a different form for the rest of them. In broad terms, the first form is for the first and second PERSONS and the second form for the third PERSON. To be more specific, this irregular group can be subdivided into 4 subgroups:

a.i. The So. and Df. V-Patterns 1, 9, 12, & 14 with the NGP values 1, 2 (M, F) S, D, P; 3 (F) P in the active and where applicable the passive VOICE.

a.ii. The So. and Df. V-Patterns 2, 10, 13, & 15 with the NGP values 3 (M, F) S, D; 3 (M) P in the active and where applicable the passive VOICE.

b.i. The Df. V-Pattern 20 with all SPs except 3 (M) P in the active and where applicable the passive VOICE.

b.ii. The Df. V-Pattern 21 with 3 (M) P in the active VOICE.

c.i. The Df. V-Patterns 16, 22, & 29 with the NGP values 1, 2 (M, F) S, D, P; 3 (M) D; 3 (F) P in the active VOICE.

c.ii. The Df. V-Patterns 17, 23, & 30 with the NGP values 3 (M, F) S; 3 (F) D; 3 (M) P in the passive VOICE.

d.i. The Df. V-Patterns 18, 24, & 31 with all SPs except 3 (M) P in the passive VOICE.

d.ii. The Df. V-Patterns 19, 25, & 32 with 3 (M) P in the passive VOICE.

II.1.2.2 The Imperfect Pattern Forms

Table 12 below shows the distribution of imperfect pattern forms for the different Subject Pronouns.

	REFERENCE in App. A, § A.	MPACT	MPPAS	NGP AND MOOD CONTEXT
1	a.(1).	acucc	ucacc	@ in I, S, J except ↓
2		acocuc	ucocac	2, 3 (F) P in I, S, J.
3	a.(2).	acacc	N/A.	@ in I, S, J except ↓
4		acocac		2, 3 (F) P in I, S, J.
5	a.(3, 7).	acocic	ucocac	@ in I, S, J.
6	a.(4).	acocuc	N/A.	@ in I, S, J.
7	a.(5).	acocuc	ucocac	@ in I, S, J.
8	a.(6, 8).	acocac	ucocac	@ in I, S, J.
9	b.(1).	ucaccic	ucaccac	@ in I, S, J.
10	c.(1).	acic	uwcac	@ in I, S, J.
11	c.(2).	awocuc	N/A.	@ in I, S, J.
12	c.(3).	N/A.	N/A.	
13	c.(4).	acowac	N/A.	@ in I, S, J.
14	c.(5).	acuwc	N/A.	@ in I, S, J except ↓
15		acuc		2, 3 (F) P in I, S, J & 1, 3 (M, F) S; 1 (M, F) D, P; 2 (M) S in J.
16	c.(6).	aciyc	N/A.	@ in I, S, J except ↓
17		acic		2, 3 (F) P in I, S, J & 1, 3 (M, F) S; 1 (M, F) D, P; 2 (M) S in J.
18	c.(7).	acociy		2, 3 (F) P; 2, 3 (M, F) D in I, S, J. 2 (F) S in I & 1 (M, F) S, D, P; 2, 3 (M) S; 3 (F) S in S.
19		acoc		2, 3 (M) P in I, S, J. 1 (M, F) S, D, P; 2, 3 (M) S; 3 (F) S in I, J & 2 (F) S in S, J.
20			ucocay	2, 3 (F) P; 2, 3 (M, F) D; 2 (F) S in I, S, J.

TABLE 12: Imperfect Pattern Forms

21			ucoc	1 (M, F) S, D, P; 2, 3 (M) S, P; 3 (F) S in I, S, J.
22	c.(8).	acocay	ucocay	2, 3 (F) P; 2, 3 (M, F) D; 2 (F) S in I, S, J.
23		acoc	ucoc	1 (M, F) S, D, P; 2, 3 (M) S, P; 3 (F) S in I, S, J.
24	c.(9).	acocay	N/A.	2, 3 (F) P; 2, 3 (M, F) D; 2 (F) S in I, S, J.
25		acoc		1 (M, F) S, D, P; 2, 3 (M) S, P; 3 (F) S in I, S, J.
26	c.(10).	acocuw		2, 3 (F) P; 2, 3 (M, F) D in I, S, J. & 1 (M, F) S, D, P; 2, 3 (M) S; 3 (F) S in S.
27		acoc		1 (M, F) S, D, P; 2, 3 (M) S; 3 (F) S in I, J & 2 (F) S in I, S, J.
28			ucocay	2, 3 (F) P; 2, 3 (M, F) D; 2 (F) S in I, S, J.
29			ucoc	1 (M, F) S, D, P; 2, 3 (M) S, P; 3 (F) S in I, S, J.
30	d.(1).	uwaccic	uwaccac	@ in I, S, J.
31	d.(2).	ucayyic	ucayyac	@ in I, S, J.
32	d.(3).	ucawwic	ucawwac	@ in I, S, J.
33	d.(4).	ucacciy		2, 3 (F) P; 2, 3 (M, F) D in I, S, J. 2 (F) S in I & 1 (M, F) S, D, P; 2, 3 (M) S; 3 (F) S in S.
34		ucacc		2, 3 (M) P in I, S, J. 1 (M, F) S, D, P; 2, 3 (M) S; 3 (F) S in I, J & 2 (F) S in S, J.
35			ucaccay	2, 3 (F) P; 2, 3 (M, F) D; 2 (F) S in I, S, J.
36			ucacc	1 (M, F) S, D, P; 2, 3 (M) S, P; 3 (F) S in I, S, J.

TABLE 12: Imperfect Pattern Forms (Contd)

OBSERVATIONS:

The forms shown in Table 12 divide into two main groups:

1) A regular group including the So.Vs 5, 6, 7, 8, & 9 above and the Df.Vs 10, 11, 13, 30, 31, & 32. In this group, only one V-Pattern form is used for all SPs in I, S, J in the active and passive VOICE.

2) An irregular group that has two different pattern forms showing different contrasts of NGP, MOOD, and VOICE. This group can be subdivided into 5 subgroups:

a.i. The So. V-Patterns 1 & 3 with all SPs except 2, 3 (F) P in I, S, J in the active and passive VOICE.

a.ii. The So. V-Patterns 2 & 4 with the NGP values 2, 3 (F) P in I, S, J in the active and passive VOICE.

b.i. The Df. V-Patterns 14 & 16 with all SPs except those covered by 15 & 17 below. This applies to the active and where appropriate the passive VOICE.

b.ii. The Df. V-Patterns 15 & 17 with the NGP values 2, 3 (F) P in I, S, J; 1, 3 (M, F) S; 1 (M, F) D, P; 2 (M) S in J in the active and where applicable the passive VOICE.

c.i. The Df. V-Patterns 18 & 33 with the NGP values 2, 3 (F) P; 2, 3 (M, F) D in I, S, J; 2 (F) S in I; 1 (M, F) S, D, P; 2, 3 (M) S; 3 (F) S in S in the active VOICE only.

c.ii. The Df. V-Patterns 19 & 34 with the NGP values 2, 3 (M) P in I, S, J; 1 (M, F) S, D, P; 2, 3 (M) S; 3 (F) S in I, J & 2 (F) S in S, J in the active VOICE only.

d.i. The Df. V-Patterns 20, 22, 24, 28, & 35 with the NGP values 2, 3 (F) P; 2, 3 (M, F) D; 2 (F) S in I, S, J.

d.ii. The Df. V-Patterns 21, 23, 25, 29, & 36 with the NGP values 1 (M, F) S, D, P; 2, 3 (M) S, P; 3 (F) S in I, S, J. The V-Patterns 20, 21, 28, 29, 35, & 36 are for the passive VOICE only and the V-Patterns 22, 23, 24, & 25 are for the active and where applicable the passive VOICE.

e.i. The Df. V-Pattern 26 with the NGP values 2, 3 (F) P; 2, 3 (M, F) D in I, S, J; 1 (M, F) S, D, P; 2, 3 (M) S; 3 (F) S in S in the active VOICE only.

e.ii. The Df. V-Pattern 27 with the NGP values 2 (F) S in I, S, J; 2, 3 (M) S; 1 (M, F) S, D, P; 3 (F) S in I, J in the active VOICE only.

II.1.3 Identification and Distribution of the V-Roots

Having identified the above abstract perfect and imperfect patterns and defined their boundaries, we can now turn our attention to carrying out the third task listed above as necessary for

the formulation of an adequate generalization about CFs. In particular, within the realization of a given pattern, we need to be able to identify valid lexical V-Roots. Further, we need to arrive at a canonical form of these roots in order to obviate the need for listing roots more than once in the database. We have argued (in Ch. 1, § II.3) that the obvious method to achieve this objective is to factor out the elements that are most variable within a given input CF. Remember that we had already stripped off the affixes, so we do not need to take account of their constituent elements. If we then take out the vowels inside the pattern, this should leave us with a *raw*, or initial, root form that we can then work on to arrive at a canonical form. Applying this method to the CFs listed (in App. A, § A), we can obtain the following Basic and Augmented root configurations noting that not all Basic roots have Augmented counterparts.

TYPE	ROOT FORM
BASIC INITIAL FORMS (in App. A, § A, a.)	d/nn jdd ɛmm :mm h-sb h-sn krm kbr kt-r s;gr qs;r jml fqr t-mn svrf ktb svkr drs frsv qtl rbɛ skn s;dq nfd- jbl h-mr zrq fth- mnh- mnɛ d-hb xd;r jls ksr d;rb nzl h-ml qlm svbk ɛsvr sds sbɛ t-lt- tsɛ xms :lf s;fr h-dq fhm ɛlm qdm svrb rkb mr: sɛd rjl.
CORRESPONDING AUGMENTED ALTERNATIVE FORMS (in App. A, § A, b.)	jddd ɛmmm :mmm h-ssn krrm kbbr kt-t-r s;ggr qs;s;r jmml fqqr t-mmn svrrf kttb drrs frrsv qttl rbbɛ skkn s;ddq nffd- jbbl h-mmr ftth- xd;d;r jlls kssr d;rrb nzzl h-mml qllm svbbk ɛsvsvr sdds sbbɛ t-llt- xmms :llf s;ffr h-ddq fhhm ɛllm svrrb rkkb mrr: rjjl.

TABLE 13: Sound Root Forms

II.2 HOMONYMY

Shape, or *how things look*, are a matter of vital importance in any recognition procedure since such procedures rely on this very shape alone to identify and detect the possible derivational origins and features of inputwords. In studying the shape, or structure, of Arabic Verbs (in App. A, § A), we attempted to collapse common forms of patterns and affixes in order to reduce to a minimum the amount of redundancy in their abstract representations and increase their generality. For instance, in Table 11, the active patterns 1 & 5: 'cacac' are the *same* and the passive patterns 1, 3 & 5: 'cucic' are the same. In Table 12, the active patterns 2, 6, & 7: 'acocuc' and the passive patterns 2, 5, 7, & 8: 'ucocac' are the same. These patterns are the same in that they have the same shape and share the same TENSE and VOICE values.

However, several of the forms under discussion: patterns, affixes, and CFs exhibit ambiguities in their NGP, MOOD, and VOICE features. Further certain CFs exhibit structural

BASIC FORMS		AUGMENTED FORMS	
INITIAL	ALTERNATIVE	INITIAL	ALTERNATIVE
wld	ld	wlld	
wrd	rd	wrrd	
wh-d	h-d	wh-h-d	
wsm		wssm	
lys	ls	N/A.	
:wl		:wwl	
k|n	kwn, kn	kwwn	
q|m	qwm, qm	qwwm	
t;|l	t;wl, t;l	t;wwl	
b|b	bwb, bb	bwwb	
j|d	jwd, jd	jwwd	
x|l	xwl, xl	xwwl	
s|:	sw:, s:	sww:	
s|d	swd, sd	swwd	
s;|r	s;yr, s;r	s;yyr	
b|t	byt, bt	byyt	
b|d;	byd;, bd;	byyd;	
bny	bn	bnny	bnn
t-ny	t-n	t-nny	t-nn
m:y	m:	N/A.	
d-ky	d-k	d-kky	d-kk
gny	gn	gnny	gnn
svqy	svq	N/A.	
:xw	:x	N/A.	
:bw	:b	N/A.	
nsw	ns	N/A.	

TABLE 14: Defective Root Form

ambiguities in that they can be SEGMENTed in more than one valid way. For instance, the Defective CF 'acocuwna' can be SEGMENTed as 'acoc$uwna' or as 'acocuw$na'. This applies to the Df. roots: {:b, :x, ns} which have the active imperfect patterns: {acoc, acocuw}.

II.2.1 Suffix Homonymy across TENSE and MOOD Values

Starting with affixes, we notice that the distribution of NGP and MOOD values in Tables 9 and 10 is not on a unique basis for each affix. Only a handful of the suffixes in fact have unique feature values while several of them show homonymic distribution: 1) between perfect and imperfect TENSE, and 2) between indicative and jussive or between subjunctive and jussive. Some disambiguation is possible using the prefixes or their absence as a context. For instance:

'ona' : 3 (F) P in I, S, J / [ ] — <y>$P/R, or / [ ] — Ø$P/R;
2 (F) P in I, S, J / [ ] — <t>$P/R; (where P/R is the realization of a pattern/root combination).

The ambiguity arises because of the Df. V-Patterns: 'acocay' followed by 'ona'. The complete CF: 'ucocayona' can then be SEGMENTed as 'ucoc$ayona' or as 'ucocay$ona'. This is applicable for the Df. root set: {svq, gn, d-k, m:} conjugated in the active VOICE and the Df. set: {t-n, t-nn, bn, bnn, gn, gnn, :x, m:} in the passive VOICE. This ambiguity reoccurs when 'na' is deleted, viz., the CF 'ucocayo' can be SEGMENTed as 'ucoc$ayo' or as 'ucocay$o' for the same set of roots and the same VOICE distribution.

Another example is 'a|':

'a|': 3 (M) D / [ ] — Ø$P/R.
3 (M) S / [ ] — Ø$P/R where the V-Root is in the set: {:b, :x, ns}.
2, 3 (M, F) D / [ ] — {y/t}$P/R.

Table 15 gives a summary of perfect and imperfect suffix homonymy across TENSE and MOOD values.

II.2.2 Perfect Pattern Homonymy across NGP and VOICE Values

The distribution of the perfect pattern forms for SPs is not always on a unique one-to-one basis. Among the 32 patterns listed in Table 11, four exhibit homonymy in NGP distribution. Of these four one pattern: 'cuc' is homonymic for VOICE as it could be active or passive. Table 16 below shows perfect pattern homonymy for NGP and VOICE values. Numbers refer to the position of particular pattern forms in Table 11.

SUFFIX FORM	NGP VALUES		MOOD VALUES	
	PERFECT	IMPERFECT	REGULAR VERBS	IRREGULAR VERBS
uwna	N/A.	2, 3 (M) P.	I.	I.
uwl	3 (M) P.	2, 3 (M) P.	S, J.	S, J.
awona	N/A.	2, 3 (M) P.	N/A.	I.
awol	3 (M) P.	2, 3 (M) P.	N/A.	S, J.
ona	3 (F) P.	2, 3 (F) P.	I, S, J.	I, S, J.
		2 (F) S.	N/A.	I.
na	3 (F) P.	2, 3 (F) P.	I, S, J.	N/A.
		3 (M) P.	N/A.	I.
al	3 (M) S.			
	3 (M) D.	2, 3 (M, F) D.	S, J.	S, J.
ay	3 (M) S.	1 (M, F) S, D, P;	N/A.	I, S.
		2, 3 (M) S; 3 (F) S.		
iy	N/A.	1 (M, F) S, D, P;	N/A.	I.
		2, 3 (M) S; 3 (F) S.		
		2 (F) S.	S, J.	S, J.
o	N/A.	1 (M, F) S, D, P;	J.	J.
		2, 3 (M) S; 3 (F) S.		
		2 (F) S.	N/A.	S, J.
a	3 (M) S.	1 (M, F) S, D, P;	S.	S, J.
		2, 3 (M) S; 3 (F) S.		
i	N/A.	1 (M, F) S, D, P;	N/A.	J.
		2, 3 (M) S; 3 (F) S.		
u	N/A.	1 (M, F) S, D, P;	I.	J.
		2, 3 (M) S; 3 (F) S.		

TABLE 15: Suffix Homonymy across TENSE and MOOD Values

II.2.3 Imperfect Pattern Homonymy across NGP, MOOD, and VOICE Values

Imperfect pattern forms are not distributed across SPs on a one-to-one basis either. Certain of these, typically Df. V-Patterns, show homonymy for NGP, MOOD, and VOICE. Among the 36 patterns listed in Table 12, four show such homonymy. The fourth one of these: 'ucacc' is ambiguous between active and passive VOICE. The specification of certain suffixes, typically irregular Df. V-Suffixes, allows the disambiguation of the VOICE value in a given homonymic pattern. For instance:

PATTERN NUMBER	PATTERN	NGP VALUES	VOICE
2	cacc	3 (M, F) S, D. 3 (M) P.	act
30		3 (M, F) S. 3 (F) D. 3 (M) P.	act
9	cac	1, 2 (M, F) S, D, P. 3 (F) P.	act
17, 23		3 (M, F) S. 3 (F) D. 3 (M) P.	act
21		3 (M) P.	act
2	cucc	3 (M, F) S, D. 3 (M) P.	pas
32		3 (M) P.	pas
12	cuc	1, 2 (M, F) S, D, P. 3 (F) P.	act
19, 25		3 (M) P.	pas

TABLE 16: Perfect Pattern Homonymy across NGP and VOICE Values

- 'ucacc' / — [ ] 'iy': active VOICE.
- 'ucacc' / — [ ] 'ay': passive VOICE.

Table 17 summarizes imperfect pattern homonymy for NGP, MOOD, and VOICE values. Numbers given there refer to the position of particular pattern forms in Table 12. The column listing *typical suffixes* shows the relevant suffixes for disambiguation. The suffixes not listed here are as in Table 12.

II.2.4 Imperfect CF Homonymy across V-Types, NGP, MOOD, and VOICE Values

Using affix context for disambiguation is not always possible. This is because certain Imperfect CFs are shared among SPs so that the distribution of these CFs for SPs is not on a unique one-to-one basis. Regular Verbs show homonymy for a fixed group of CFs, typically between second and third PERSON for the singular and dual. For example:

a. t$acocac${u/a/o} / [ ] regular root: 2 (M) S, 3 (F) S.

b. t$acocac${a|ni/a|} / [ ] regular root: 2 (M, F) D, 3 (F) D.

Graphological changes in the suffix forms of irregular Df.Vs give rise to further ambiguities beside *a.* and *b.* above. For instance:

tacoc$ayona / [ ] irregular Defective root: 2 (F) S, P.

PATTERN NUMBER	PATTERN	NGP AND MOOD CONTEXT	VOICE	TYPICAL SUFFIX
10	acic	@ in I, S, J.	act	
17		2, 3 (F) P in I, S, J & 1, 3 (M, F) S; 1 (M, F) D, P; 2 (M) S in J.	act	
19	acoc	2, 3 (M) P in I, S, J. 1 (M, F) S, D, P; 2, 3 (M) S; 3 (F) Sin I, J & 2 (F) S in S, J.	act	iy
23, 25		1 (M, F) S, D, P; 2, 3 (M) S, P; 3 (F) S in I, S, J.	act	ay
27		1 (M, F) S, D, P; 2, 3 (M) S; 3 (F) S in I, J & 2 (F) S in I, S, J.	act	uw
21, 23, 29	ucoc	1 (M, F) S, D, P; 2, 3 (M) S, P; 3 (F) S in I, S, J.	pas	ay
1	ucacc	@ in I, S, J except ↓ 2, 3 (F) P in I, S, J.	pas	
36		1 (M, F) S, D, P; 2, 3 (M) S, P; 3 (F) S in I, S, J.	pas	ay
34		2, 3 (M) P in I, S, J. 1 (M, F) S, D, P; 2, 3 (M) S; 3 (F) S in I, J & 2 (F) S in S, J.	act	iy

TABLE 17: Imperfect Pattern Homonymy across NGP, MOOD, and VOICE Values

The modification of 'iyna', 2 (F) S to 'ayona' results in the CF becoming SEGMENTable in two different but valid ways since both 'acoc' and 'acocay' are valid patterns for this group of roots. The SEGMENTation can be performed as follows:

c. t$acoc$ayona: 2 (F) S. d. t$acocay$ona: 2 (F) P.

Further, these ambiguities in irregular Df.Vs arise for either active or passive VOICE but not for both, so that NGP and MOOD homonymy for these Verbs does not coincide with VOICE homonymy. Table 18 summarizes homonymy in CFs for NGP, MOOD, and VOICE features. Note that in 1 to 4 below homonymy applies for any of the suffixes specified: <u> and 'a|ni' are for the indicative, <a> for the subjunctive, <o> for the jussive, and 'a|' for the subjunctive and jussive. For CFs 5 to 12 feature homonymy is compounded with pattern ambiguity: each of these CFs can be SEGMENTed in two different but valid ways. Table 18 lists these SEGMENTations together. In CFs 8, 9, & 10, the ambiguity arises because of the

allomorph 'na' of the feminine plural morpheme 'ona', since the allomorphs are in free variation, there is nothing so far, to stop the generation of the ungrammatical SEGMENTations. These are not allowed, as <o> can be deleted only if the characters at either side of it are identical. We will have to find a solution to enforce this condition at some stage in the analysis. CFs 11 & 12 have no feature homonymy but still have two possible valid SEGMENTations because of pattern ambiguity. Note that, for CFs 1 to 4, <v> represents any of the vowel set: {u, a, i}; and A/P is active/passive.

II.2.5 Root Homonymy across Pattern Distribution and TRANSITIVITY Values

Among the roots listed in § I.2 above, there are two roots which do not have unique distinguishing patterns. While the other roots have a unique association with their patterns for each VOICE/TENSE form, the roots: {h-sb, t-mn} associate with more than one pattern for each of the 4 classes of pattern: active (perfect, imperfect) and passive (perfect, imperfect). Further, these roots are homonymic for TRANSITIVITY in that each different pattern in each of the classes they have indicates a different TRANSITIVITY value. This is because these roots are homonymic in "meaning", viz., 'h-sb' is "to count", and also "to think, consider", and 't-mn' is "to become expensive", or "to join a group of 7 and make them 8". Some disambiguation of which value is meant is possible using the penultimate vowel of the pattern in question. Thus:

a.i. 'h-sb' / [ ] — >2 (pattern) = {a/u} & VOICE = act $\Longrightarrow$ 'h-sb' is monotransitive.

a.ii. 'h-sb' / [ ] — >2 (pattern) = <i> & VOICE = act $\Longrightarrow$ 'h-sb' is xtransitive.

a.iii. 'h-sb' / [ ] — VOICE = pas $\Longrightarrow$ 'h-sb' is nctransitive.

b.i. 't-mn' / [ ] — >2 (pattern) = {a/i} & VOICE = act $\Longrightarrow$ 't-mn' is monotransitive.

b.ii. 't-mn' / [ ] — >2 (pattern) = <u> $\Longrightarrow$ 't-mn' is intransitive.

b.iii. 't-mn' / [ ] — VOICE = pas $\Longrightarrow$ 't-mn' is intransitive.

We can see that rule *a.iii.* is ambiguous for TRANSITIVITY while rule *b.iii.* is not ambiguous as 't-mn' in its second meaning is not passivizable. Table 19 below shows root homonymy across pattern distribution and TRANSITIVITY values.

TYPE	IMPERFECT CF	NGP AND MOOD VALUES	VOICE	ROOTS AFFECTED
REGULAR	1. :$vcocvc$ {u/a/o}	1 (M, F) S in I, S, J.	A/P	all roots.
	2. n$vcocvc$ {u/a/o}	1 (M, F) D, P in I, S, J.	A/P	all roots.
	3. t$vcocvc$ {u/a/o}	2 (M) S; 3 (F) S in I, S, J.	A/P	all roots.
	4. t$vcocvc$ {a|ni/a|}	2 (M, F) D; 3 (F) D in I, S, J.	A/P	all roots.
IRREGULAR				
	5. t$ucoc$ayona	2 (F) S in I.	pas	{:x, m:, bn, t-n}.
	t$ucocay$ona	2 (F) P in I, S, J	pas	{:x, m:, bn, t-n}.
	6. t$ucacc$ayona	2 (F) S in I.	pas	{bnn, t-nn, gnn}.
	t$ucaccay$ona	2 (F) P in I, S, J	pas	{bnn, t-nn, gnn}.
	7. t$acoc$ayona	2 (F) S in I.	act	{m:, d-k, svq, gn}.
	t$acocay$ona	2 (F) P in I, S, J	act	{m:, d-k, svq, gn}.
	8. y/t$acoc$uwna	2, 3 (M) P in I.	act	{:b, :x, ns}.
	*y/t$acocuw$na	2, 3 (F) P in I, S, J.	act	{:b, :x, ns}.
DEFECTIVE	9. t$acoc$iyna	2 (F) S in I.	act	{bn, t-n}.
	*t$acociy$na	2 (F) P in I, S, J	act	{bn, t-n}.
	10. t$ucacc$iyna	2 (F) S in I.	act	{bnn, t-nn, gnn, d-kk}.
	*t$ucacciy$na	2 (F) P in I, S, J.	act	{bnn, t-nn, gnn, d-kk}.
	11. t$acoc$ayo	2 (F) S in S, J	act	{m:, d-k, svq, gn}.
	t$acocay$o	2 (F) S in S, J.	act	{m:, d-k, svq, gn}.
	12. t$ucacc$ayo	2 (F) S in S, J.	pas	{bnn, t-nn, gnn}.
	t$ucaccay$o	2 (F) S in S, J.	pas	{bnn, t-nn, gnn}.

TABLE 18: Imperfect CF Homonymy across V-Types, NGP, MOOD, and VOICE Values

ROOT	PERFECT PATTERN		IMPERFECT PATTERN		TRANSI-	PENUL-
	ACTIVE	PASSIVE	ACTIVE	PASSIVE	TIVITY	TIMATE VOWEL
h-sb	cacac		acocuc		mtr	{a/u}
		cucic		ucocac	ntr	{a/i}
	cacic		acocic		xtr	<i>
		cucic		ucocac	ctr	{a/i}
t-mn	cacac		acocic		mtr	{a/i}
		cucic		ucocac	ntr	{a/i}
	cacuc		acocuc		ntr	<u>
		Ø		Ø		

TABLE 19: Root Homonymy across Pattern Distribution and TRANSITIVITY Values

II.2.6 Categorial Homonymy for Simple CFs

Certain lexical CFs, or CF realizations, show categorial homonymy. Typically, lexical active Perfect CFs in 3 (M) S are homonymic between Verb CATEGORY, noted as *VB* and Nominal Modifier CATEGORY noted as *MD*. The lexical CF 'εasvara' can be a Verb or a Numeral Modifier noted as *MM*. In addition, certain lexical active Imperfect CFs are ambiguous between Verb CATEGORY and adjectival Modifier CATEGORY also noted as *MD*. Note that *CAT1* refers to the initial categorial interpretation and *CAT2* refers to alternative interpretations. A summary of categorial homonymy for lexical Simple CFs is given in Table 20 below which also gives a listing of the features involved in categorial homonymy as well as the roots and patterns affected. Clearly, these forms will have different features depending on which categorial interpretation they receive. The features and forms themselves could be used in raising flags to signal the specific ambiguity for later disambiguation. Note that, at this point, no disambiguation is possible since the homonymy involves the whole CF, i.e., the pattern realization and the affixes. Hence, we could not for example use affix context to resolve the categorial ambiguities of these CFs. In particular, the coincidence of the indicative-MOOD marker <u> with the nominative-CASE marker <u> and of the subjunctive-MOOD marker/perfect suffix <a> with the accusative-CASE marker <a> reinforces rather than alleviates the ambiguity in lexical Simple CFs.

LEXICAL CF	PATTERN	ROOT	CAT1	VERB FEATURES	CAT2	NOMINAL FEATURES
'ɛasvara' "he joined 10 or 10"	cacac	ɛsvr	VB	3 (M) S, prf, act, mtr.	MM	3 (M) S, svm, acm, ndf, 10.
'd-ahaba' "he went or gold"	cacac	d-hb	VB	3 (M) S, prf, act, ptrl.	MD	3 (M) S, svm, acm, ndf.
'h-asaba' "he counted or honour"	cacac	h-sb	VB	3 (M) S, prf, act, mtr.	MD	3 (M) S, svm, acm, ndf.
't-amana' "he joined 8 or cost"	cacac	t-mn	VB	3 (M) S, prf, act, mtr.	MD	3 (M) S, svm, acm, ndf.
'jabala' "he created or mountain"	cacac	jbl	VB	3 (M) S, prf, act, mtr.	MD	3 (M) S, svm, acm, ndf.
'qalama' "he cut or pen"	cacac	qlm	VB	3 (M) S, prf, act, mtr.	MD	3 (M) S, svm, acm, ndf.
'walada' "he gave birth or son"	wacac	ld	VB	3 (M) S,, prf, act, mtr.	MD	3 (M) S, svm, acm, ndf.
'ɛamma' "it spread or paternal uncle"	cacc	ɛmm	VB	3 (M) S, prf, act, ntr.	MD	3 (M) S, svm, acm, ndf.
'jadda' "he cut or grandfather"	cacc	jdd	VB	3 (M) S, prf, act, mtr.	MD	3 (M) S, svm, acm, ndf.
'xa|la' "he managed or maternal uncle"	ca|c	xl	VB	3 (M) S, prf, act, mtr.	MD	3 (M) S, svm, acm, ndf.
'ba|ba' "he made an opening or door"	ca|c	bb	VB	3 (M) S, prf, act, ntr.	MD	3 (M) S, svm, acm, ndf.
':axod;aru' "I became green or green"	acocac	xd;r	VB	1 (M, F) S, imp, act, ind, ntr.	MD	3 (M) S, svm, nmm, ndf.
' :axod;ara' "I became green or green"	acocac	xd;r	VB	1 (M, F) S, imp, act, sub, ntr.	MD	3 (M) S, svm, acm, ndf.

TABLE 20: Categorial Homonymy for Simple CFs

II.3 GRAPHOTACTIC AFFIXATION CONDITIONS

Besides the problems of homonymy involved in distinguishing particular affixes, patterns, roots, and CFs, another class of problem hinted at above is that of the *graphotactic* variation involved at morpheme boundaries. This is because the affixation of bound morphemes to other morphemes is subject to *Graphotactic Conditions* which affect their graphic shape. In particular, there are graphotactic sequential constraints governing the affixation of suffixes to CFs such that whenever a given lexical pattern ends in <n> and a suffix following it starts with the mute <o> followed by <n> then <o> is deleted. This also applies to <t>: the mute <o> of

a suffix is deleted for every lexical pattern that ends with <t> where <o> is followed by <t>. In other circumstances, <o> cannot be deleted. For instance:

*kunona $\rightarrow\!\!\!\!\rightarrow$ kunna; *tad/onunona $\rightarrow\!\!\!\!\rightarrow$ tad/onunna; *bitotu $\rightarrow\!\!\!\!\rightarrow$ bittu;
*waladna $\rightarrow\!\!\!\!\rightarrow$ waladona; *yadorusna $\rightarrow\!\!\!\!\rightarrow$ yadorusona.

Since this affects both perfect and imperfect, active and passive Verbs we can generalize consonantal sequence rules of the type in *a.* and *b.* below where *b.* is the inverse of *a.*

a. $*c_1oc_1$ but c_1c_1 where $c_1 = c_1$.

b. $*c_1c_2$ but c_1oc_2 where $c_1 \neq c_2$.

Further, the selection of a given suffix allomorph is governed by such criteria as root type, TENSE, and VOICE. In particular, the choice of a *Defective suffix* variant depends on the lexical root and the type of pattern involved. We define *Defective suffixes* as this set: {i, ay, uw, ayona, awona, awo|} and these suffixes with the specified values: 'a|' with 3 (M) S, prf; 'iy' with 1, 3 (M, F) S, and 2 (M) S, imp; <i> and <u> in S, J; <a> in J; and 'ayo' in S. To be more specific, all the *default set* of roots will be restricted from attaching with all Defective suffixes. We define the *default set* of roots as all the Basic and Augmented Sound roots (excepting Doubled roots), the Basic (Defective Quasi-Sound and Defective Hollow) roots, and the Augmented (Defective non-Weak) roots. We define the *exceptional set* of roots as all the Basic and Augmented Df.Wk. roots. The distribution that we can deduce from Tables 9 to 12 and Table 15 above of the Df. suffixes for the exceptional roots is as follows:

'ay' with 3 (M) S, prf, act, for the
set: {bn, t-n, m:, d-k, gnn, bnn, t-nn, d-kk};
with 1, 3 (M, F) S, D, P & 2 (M) S in I, S, imp, act, for the
set: {m:, d-k, gn, svq}; and
with 1, 3 (M, F) S, D, P & 2 (M) S in I, S, imp, pas, for the
set: {:x, :b, ns, bn, t-n, m:, d-k, gnn, bnn, t-nn}.

'a' with 1, 3 (M, F) S, D, P & 2 (M) S in J, imp, act, for the
set: {m:, d-k, gn, svq};
with 1, 3 (M, F) S, D, P & 2 (M) S, imp, pas, for the
set: {:x, :b, ns, bn, t-n, m:, d-k, gnn, bnn, t-nn}; and
with 1, 3 (M, F) S, D, P & 2 (M) S in J imp, act & pas, for the
set: {d/nn, jdd, :mm, ɛmm}.

'uw' with 1, 3 (M, F) S, D, P & 2 (M) S in I, imp, act, for the
set: {:x, :b, ns}.

'u' with 1, 3 (M, F) S, D, P & 2 (M) S in J, imp, act, for the
set: {:x, :b, ns}.

'iy' with 1, 3 (M, F) S, D, P & 2 (M) S in I, imp, act, for the
set: {bn, t-n, gnn, bnn, t-nn, d-kk}.

'i' with 1, 3 (M, F) S, D, P & 2 (M) S in J, imp, act, for the
set: {bn, t-n, gnn, bnn, t-nn, d-kk}.

'a|' with 3 (M) S, prf, act, for the set: {:b, :x, ns}.

'ayo' with 2 (F) S in S, J, imp, act, for the
set: {gn, svq, m:, d-k}; and
with 2 (F) S in S, J, imp, pas, for the
set: {bn, t-n, m:, d-k, :x, :b, ns, bnn, t-nn, gnn}.

'awo|' with 2, 3 (M) P in S, J, imp, pas, for the
set: {:b, :x, ns, bn, t-n, m:, d-k, gnn, bnn, t-nn, d-kk}; and
with 3 (M) P, prf, act, for the
set: {bn, t-n, gn, svq, m:, d-k}.

'awona' with 2, 3 (M) P in I, imp, pas, for the
set: {:b, :x, ns, bn, t-n, m:, d-k, gnn, bnn, t-nn, d-kk}.

'ayona' with 2 (F) S in I, imp, act, for the set: {gn, svq, m:, d-k}; and
with 2 (F) S in I, imp, pas, for the
set: {bn, t-n, m:, d-k, :x, :b, ns, bnn, t-nn, gnn}.

Note that there is a modicum of unavoidable redundancy in these statements. This can be justified on account of the fact that the distribution of the above Df. suffixes, for the sets of exceptional roots, is not on a unique one-to-one basis; rather, several of the root sets overlap in this distribution. Furthermore, the values correlated with the suffix-root groups distinguished above, and specified for such properties as NGP, MOOD, and VOICE, are not distributed on a clear-cut or unique basis. However, the VOICE specifications above apply in the case of the passive only, if the root is passivizable. Roots that are not passivizable are indicated (in App. A, § A) with the abbreviation N/A. Note also that we define the *default suffixes* as all those that are not in the Df. suffixes, and we can say that for all values other than those specified above all default roots and exceptional roots are allowed to attach freely with the default suffixes with the values specified in Tables 9 to 12. These rules allow us to exclude such ungrammatical sequences as the following CFs with regular patterns but Df. suffixes: '*yad;orib$i', "*he hits", '*wulid$awo|', "*they were born", '*daras$ay', "*he studied", '*yajolis$uw', "*he sits", '*yad-ohab$iy', "*he goes", '*tasvorab$ayona', "*you drink", and '*yuqotal$awona', "*they are killed".

We can generalize the above sequence conditions which we will call *Affix Selection Rules* (*ASRs*) as follows:

c. *P/R$S where S is a suffix, and where the root is in the default root set and S in

the Defective suffixes; but P/R$S where the root is in the default root set, and S in the non-Defective suffixes.

d. *P/R$S where the root is in the exceptional root set, and S in the Defective suffixes with values other than those specified immediately above; but P/R$S where the root is in the exceptional root set, and S in the Defective or the non-Defective suffixes under the appropriate specified conditions.

II.4 AFFIXATION OF OTHER AFFIXES TO SIMPLE CFs

We have seen that a Simple CF is a composite of a root embedded in a pattern realization which is in turn attached with one or two V-Affixes depending on whether it is in the perfect or imperfect TENSE. However, a Simple CF itself can be attached with more affixes of a different nature than those seen so far and subject to various conditions.

II.4.1 Affixation to Accusative Personal Pronouns

Simple CFs can be attached with *encliTic Pronouns* (*TPs*). In Arabic, most TPs are in fact homonymic between accusative and genitive CASE. Disambiguation of TPs is possible depending on which category of morpheme they are attached to: if they are attached to Verbs then they are *Accusative Pronouns* (*CPs*); if they are attached to non-Verbs they are *Genitive Pronouns* (*GPs*). However, the first-PERSON-singular TP has one distinct form for each of these CASES. Table 21 below lists these enclitic Pronouns in M.S.A., while Table 22 gives a four-dimensional perspective of overlap in certain of these TPs, typically in the dual form. Further ambiguity arises when certain of these TPs coincide in form with other SPs. Specifically, third-PERSON dual and plural TPs have the same form as their SP counterparts. However, SPs are free morphemes while TPs are bound ones, and this fact allows their disambiguation. Certain other TPs coincide with V-Suffix forms. For instance, the allomorph 'na|' can be a first-PERSON-plural V-Suffix as in 'kun$na|', "we were", or a first-PERSON-plural CP as in 'd;arabuw$na|', "they hit us". The Genitive Pronoun 'iy' as in 'kita|b$iy', "my book", coincides with the second-PERSON-feminine-singular form of a V-Suffix in the subjunctive or jussive as in 't$ajolis$iy', "you sit". The first is attached to a Noun, and the second to a Verb. Hence, contextual disambiguation is often possible, but homonymy still poses problems (for example, in the ASSIGNment of default property values to the lexical entries affected). Table 23 summarizes homonymy in the enclitic Pronouns across NGP, MOOD, TENSE, and CASE values.

	PRONOMINAL FORM		INITIAL		ALTERNATIVE		NGP, REF, AND DFN
	DEFAULT	CONTEXT	CAT	CASE	CAT	CASE	VALUES
1	niy		CP	acm	Ø	Ø	1 (M, F) S, exl, dfp.
2	iy	ya	Ø	Ø	GP	obm	1 (M, F) S, exl, dfp.
3	na|		CP	acm	GP	obm	1 (M, F) D, P, exl, dfp.
4	ka		CP	acm	GP	obm	2 (M) S, exl, dfp.
5	ki		CP	acm	GP	obm	2 (F) S, exl, dfp.
6	kuma|		CP	acm	GP	obm	2 (M, F) D, exl, dfp.
7	kumo		CP	acm	GP	obm	2 (M) P, exl, dfp.
8	kunna		CP	acm	GP	obm	2 (F) P, exl, dfp.
9	hu	hi	CP	acm	GP	obm	3 (M) S, ana, dfp.
10	ha|		CP	acm	GP	obm	3 (F) S, ana, dfp.
11	huma|	hima|	CP	acm	GP	obm	3 (M, F) D, ana, dfp.
12	humo	himo	CP	acm	GP	obm	3 (M) P, ana, dfp.
13	hunna	hinna	CP	acm	GP	obm	3 (F) P, ana, dfp.

TABLE 21: Enclitic Pronouns in M.S.A.

II.4.2 TRANSITIVITY Affixation Conditions

The affixation of CPs to Simple CFs is subject to TRANSITIVITY conditions. Specifically, such affixation is allowed for these classes of Verb:

a. Monotransitive, e.g., 'svariba$hu', "he drank it".

b. Ditransitive, e.g., 'manah-a$hu', "he granted him ...".

c. Xtransitive, e.g., 'h-asiba$hu', "he thought him ...".

d. Ptransitive2, e.g., 'manaʕa$hu', "he prevented him".

However, the following classes are not allowed to attach with CPs:

e. Intransitive, e.g., '*jalasa$hu', "*he sat him".

f. Ptransitive1, e.g., '*nazala$hu', "*he went down him".

g. Ptransitive, e.g., '*d-uhiba$hu', "*was gone him".

In addition, CFs with categorial homonymy as in § II.6 above are allowed to take CPs. However, note that upon affixation it becomes possible to disambiguate a subset of the homonymic CFs as follows:

h. Intransitive$TP $\Longrightarrow$ Noun$GP, e.g., 'ba\|ba$hu', "his door".

i. Ptransitive1$TP $\Longrightarrow$ Noun$GP, e.g., 'd-ahaba$hu', "his gold".

Further, homonymic CFs attached with the first-PERSON-singular CP 'niy' have to be interpreted as Verbs since only the first-PERSON-singular allomorphs {iy, ya} are attached

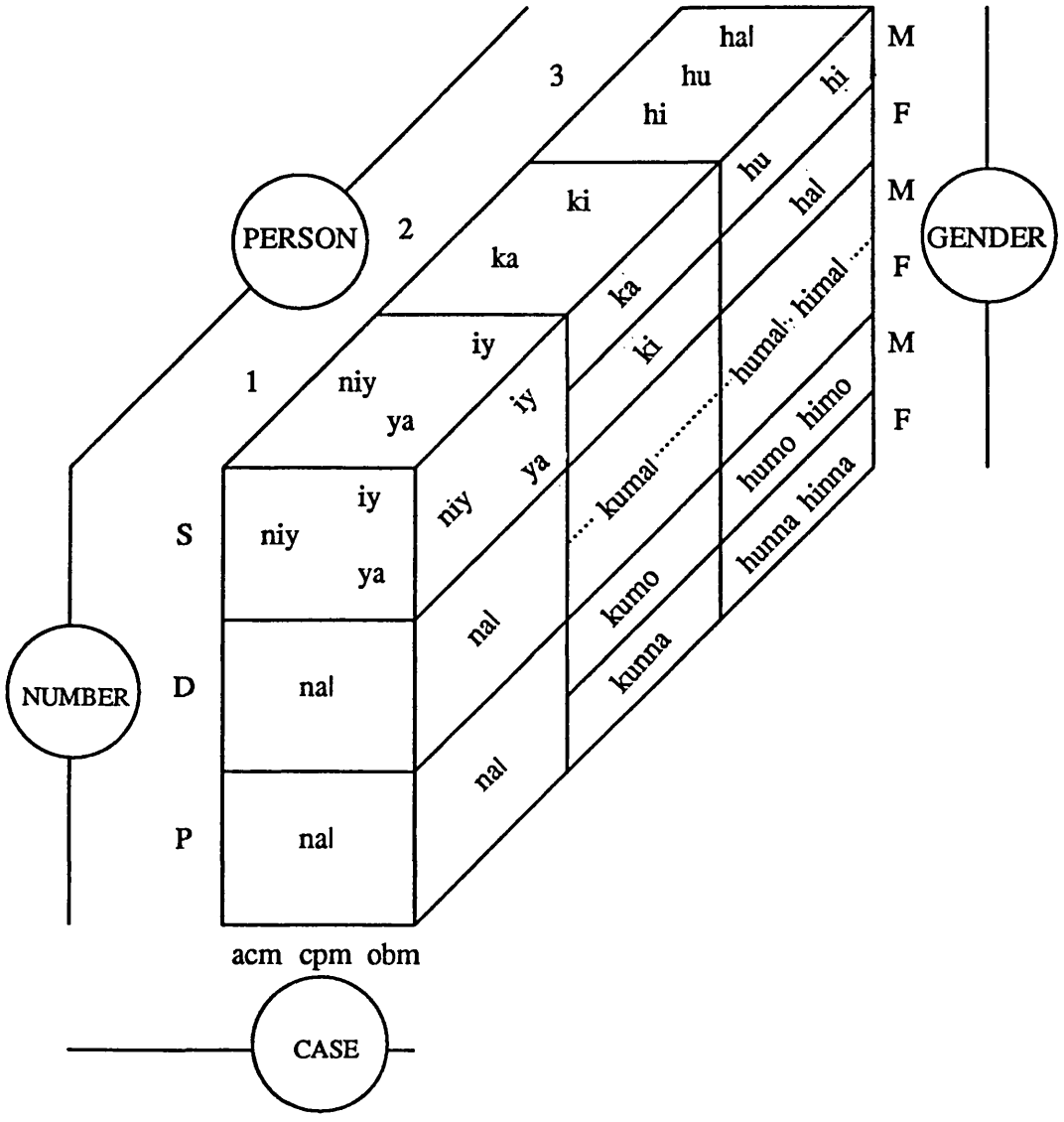


TABLE 22: Feature Dimensions for Arabic Enclitic Pronouns: Overlap

PRONOMINAL FORM	NGP AND CASE VALUES			NGP, MOOD, AND TENSE VALUES AS V-SUFFIX
	ACCUSATIVE	GENITIVE	NOMINATIVE	
iy	Ø	1 (M, F) S, obm.	Ø	2 (F) S in S, J, imp.
na|	1 (M, F) D, P, acm-obm.		Ø	1 (M, F) D, P, prf.
huma|	3 (M, F) D, P, acm-obm.		3 (M, F) D, nmm.	Ø
humo	3 (M) P, acm-obm.		3 (M) P, nmm.	Ø
hunna	3 (F) P, acm-obm.		3 (F) P, nmm.	Ø

TABLE 23: Enclitic Pronominal Homonymy across NGP, MOOD, TENSE, and CASE Values

with Nouns. Hence, 'xa\|la$niy', "he took care of me", is a VB$CP and 'xa\|l$iy', "my uncle", is a Noun followed by a GP. Table 24 gives a summary of categorial homonymy for CFs attached with TPs and disambiguation contexts where possible. Here, a Referential Complex CF, which is a Simple CF attached with a CP is given the CATEGORY or CAT1 VR, if the Simple CF is a Verb, and the CATEGORY or CAT2 AN, if the Simple CF is a Noun. The Complex CF itself is given a referential feature value *xxl* which signals an ambiguity between collocutive and exlocutive. REFERENCE is said to be *collocutive* if the TP attached is first or second PERSON, and *exlocutive* if it is third PERSON. All Nominal features are further explained in Chapter 6, § II.1.

II.4.3 Graphotactic Affixation Conditions

Besides TRANSITIVITY conditions, the affixation of Simple CFs with CPs is subject to Graphotactic Conditions which affect their graphic shape. In particular, upon the affixation of CPs to all masculine plural Simple CFs both in perfect and imperfect TENSE, in the active and passive VOICE, such CFs undergo *simple* graphemic *transformations* (henceforth, we shall use the term (*simple*) *transformation*, as opposed to a *generative transformation* (which we use in the sense defined in Ch. 1, § I.3.1), in a neutral sense, to refer to a surface form change, or modification, such as a deletion, a substitution, or an insertion). The transformations affect all the Pronouns listed in Table 20 above and the changes in question involve substitution or deletion. For instance, the Simple CF: 'd;arabotumo', "you hit", when attached with the CP 'hu', "him", becomes 'd;arabotumuw$hu', "you hit him", and the Simple CF: 'yudarrisuw\|', "they teach", when attached with the CP 'hu' becomes 'yudarrissuw$hu', "they teach him".

LEXICAL CF	-CP	CAT	CAT1	VERB FEATURES	-GP	CAT	CAT2	NOMINAL FEATURES
'ɛasvara' "he joined 10"	@	CP	VR	3 (M) S, prf, act, mtr, xxl.	*@	Ø	Ø	N/A.
'd-ahaba' "his gold"	*@	Ø	Ø	N/A.	@	GP	AN	3 (M) S, fvm, acm, dfp, xxl.
'h-asaba' "he counted it or his honour"	@	CP	VR	3 (M) S, prf, act, mtr, xxl.	@	GP	AN	3 (M) S, fvm, acm, dfp, xxl.
't-amana' "he joined 8 or its cost"	@	CP	VR	3 (M) S, prf, act, mtr, xxl.	@	GP	AN	3 (M) S, fvm, acm, dfp, xxl.
'jabala' "he created it or his mountain"	@	CP	VR	3 (M) S, prf, act, mtr, xxl.	@	GP	AN	3 (M) S, fvm, acm, dfp, xxl.
'qalama' "he cut it or his pen"	@	CP	VR	3 (M) S, prf, act, mtr, xxl.	@	GP	AN	3 (M) S, fvm, acm, dfp, xxl.
'walada' "he gave birth to him or his son"	@	CP	VR	3 (M) S, prf, act, mtr, xxl.	@	GP	AN	3 (M) S, fvm, acm, dfp, xxl.
'ɛamma' "his uncle"	*@	Ø	Ø	N/A.	@	GP	AN	3 (M) S, fvm, acm, dfp, xxl.
'jadda' "he cut or his grand-father"	@	CP	VR	3 (M) S, prf, act, mtr, xxl.	@	GP	AN	3 (M) S, fvm, acm, dfp, xxl.
'xa|la' "he took care of or uncle"	@	CP	VR	3 (M) S, prf, act, mtr, xxl.	@	GP	AN	3 (M) S, fvm, acm, dfp, xxl.
'ba|ba' "his door"	*@	Ø	Ø	N/A.	@	GP	AN	3 (M) S, fvm, acm, dfp, xxl.
':axod;aru' "green"	*@	Ø	Ø	N/A.	@	GP	AM	3 (M) S, fvm, nmm, ndf.
':axod;ara' "green"	*@	Ø	Ø	N/A.	@	GP	AM	3 (M) S, fvm, acm, dfp, xxl.

TABLE 24: Categorial Homonymy for Complex CFs

We can state the conditions described at the beginning of this section in a general form as follows:

a. $\forall$ >1 (CF) = <o> $\Longrightarrow$ <o> $\rightarrow\!\!\!\!\rightarrow$ 'uw' / [ ] CF$CP, where CF is a VB in the perfect TENSE with the NGP values 2 (M) P.

b. $\forall$ >1 (CF) = <\|> $\Longrightarrow$ <\|> $\rightarrow\!\!\!\!\rightarrow$ Ø / [ ] CF$CP, where CF is a VB in the perfect TENSE with the NGP values 3 (M) P or in the imperfect TENSE with the NGP values 2, 3 (M) P and MOOD values subjunctive or jussive.

Rule *a.* is a rule of SUBSTITUTION and should be read as follows: 'whenever the final character of a CF with the property values specified in *a.* is the mute <o>, substitute it with 'uw' upon affixation of a CP to the CF'. Rule *b.* is a rule of DELETION and should be read as follows: 'whenever the final character of a CF with the property values specified in *b.* is the character <|> delete it upon affixation of a CP to the CF'.

Furthermore, we noted in Table 21 above that certain third-PERSON TPs have two allomorphs, one where the second character is <u> and another where it is <i>. The affixation of such Pronouns as CPs to Simple CFs is not a matter of free variation but complementary distribution. In particular, the affixation of a given allomorph to a CF depends on whether the final character of the CF is *compatible* with the second character of the allomorph involved. Hence, this affixation can be said to be subject to a kind of *vowel harmony* that is motivated by pronunciation rules. According to CRYSTAL, 1980: 380, *vowel harmony* is "a term used to describe a phonological pattern in some languages, where all the vowels in a word share certain features, e.g. they are all articulated with the front of the tongue, or all are rounded". Indeed, in Arabic, we can describe a similar phenomenon such that front vowels are compatible with front vowel variants, and back vowels with back vowel variants, but not vice versa. We shall use *vowel harmony*, or *Vocalic Compatibility*, to refer to the commonness of features between vowels on the graphological level. This *Vocalic Compatibility* can be expressed formally as follows:

c. $\forall$ CF & CP / [ ] CF$CP,

i. *if* >1 (CF) = {i/y}, or

ii. *if* >1 (CF) = <o> & >2 (CF) = <y> $\Longrightarrow$ <2 (CP) = <i>.

iii. *if* >1 (CF) = <o> & >2 (CF) $\neq$ <y>, or

iv. *if* >1 (CF) = {a/u/w/|} $\Longrightarrow$ <2 (CP) = <u>.

Rule *c.* is a rule of *compatibility* which should be read as follows: "forall CPs affixed to CFs, the second character of the CP should be <i> where the final character of the CF is a member of the set: {i, y} or is equal to <o> and is preceded by <y>. It should be <u> where the final character of the CF is <o> not preceded by <y> or is a member of the set: {a/u/w/|}".

II.4.4 Affixation of Future Particles to Simple CFs

Simple CFs that may or may not be attached with Accusative Pronouns, can be attached with *futurization* Particles. Namely, the Bound Particles 'sa', "going to", and 'li', "in order to", have the effect of modifying the CF to a future aspect. While the affixation of such Particles to CFs is not subject to Graphotactic Conditions as in pronominal affixation above, it is subject to TENSE and MOOD restrictions. Specifically, Future Particles can be attached only to Imperfect CFs. The Particle 'sa' can be attached to Imperfect CFs that are in the indicative

MOOD. The Particle 'li' can be attached to Imperfect CFs that are in the subjunctive MOOD. We shall refer to these conditions as *GOVERNMENT* conditions. (We have already defined—in Ch. 1, § I.1—the term *GOVERNMENT*, we can add to that definition that we are referring here to the phenomenon whereby a given morphosyntactic CATEGORY, such as a Particle, a Preposition or a Verb, *assigns* a specific MOOD MARK to a CATEGORY: Verb or a specific CASE MARK to a CATEGORY: *Nominal*, which we define in Chapter 6, § I.1. Here, we use the term *assignment*, to refer simply to the condition whereby the governing CATEGORY, or governor, (called *'ɛa|mil'*, in Arabic) requires the governed CATEGORY, or dependent, (called *'maɛomuwl'*, in Arabic) to carry a specific MOOD or CASE MARK. Further, we use the term *GOVERNMENT* as a property for which the governor has a given value which has to be matched by the value carried by the dependent for MOOD or CASE, thus expressing a direct and simple method for the implementation of the concept of GOVERNMENT in the actual parser.)

It turns out that the MOOD GOVERNMENT conditions are useful conditions in the reinterpretation of CFs that are homonymic for MOOD values. For instance, the Simple CF 'yalidona', "they give birth", with the NGP and MOOD values: 3 (F) P in I, S, J, can in fact be disambiguated under affixation as follows:

a.i. 'yalidona' / [ ] — 'sa' $\Longrightarrow$ 3 (F) P in I.

a.ii. 'yalidona' / [ ] — 'li' $\Longrightarrow$ 3 (F) P in S.

Another example is the homonymic CF 'tubonayona', "you/they are built", which has the NGP and MOOD values: 2 (F) S in I and 2 (F) P in I, S, J, and the CF 'tubonayo', "you are built", which has the values 2 (F) S in S, J can be disambiguated under affixation as follows:

b.i. 'tubonayona' / [ ] — 'sa' $\Longrightarrow$ 3 (F) S, P in I.

b.ii. 'tubonayona' / [ ] — 'li' $\Longrightarrow$ 3 (F) P in S.

b.iii. 'tubonayo' / [ ] — 'li' $\Longrightarrow$ 3 (F) S in S.

b.iv. '*tubonayo' / [ ] — 'sa'.

It is also possible to resolve categorial homonymy at this point. We noted in Table 20, two CFs that are homonymic for CATEGORY between Verb and adjectival Modifier. When attached with Future Particles, these have only one possible interpretation as follows:

c.i. ':axod;aru' / [ ] — 'sa' $\Longrightarrow$ VB, 1 (M, F) S in I, imp, act, ntr.

c.ii. ':axod;ara' / [ ] — 'li' $\Longrightarrow$ VB, 1 (M, F) S in S, imp, act, ntr.

Note that the above conditions for TENSE and MOOD are also inhibiting conditions in the sense that they *constrain* futurization Particles from being attached to CFs that do not have

satisfactory TENSE/MOOD feature requirements as in *b.iv*. This can be generalized as follows:

d.i. *CF$'sa'/'li' / [ ] CF [+perfect].

d.ii. *CF$'sa' / [ ] CF [+imperfect] and CF [+subjunctive]/CF [+jussive].

d.iii. *CF$'li' / [ ] CF [+imperfect] and CF [+indicative]/CF [+jussive].

II.5 STRUCTURAL DEFINITION OF A COMPLEX CF

The formal specifications described in our analysis so far, allow us to deduce precise structural definitions of the morphological units under investigation in Complex CFs. These definitions should in turn facilitate the task of building an adequate algorithm for the efficient parsing of those units.

II.5.1 MS Rules for the Complex CF

We have seen in § II.1.3 that a Simple CF is a composite of a root embedded in a pattern (P/R) which is in turn attached with one affix for a *Perfect CF* (*PCF*) and with two affixes for an *Imperfect CF* (*ICF*).

We have also seen in § II.4 that a Complex CF is either a *Referential CF* (*RCF*) which is a Simple CF attached with a CP or a *Future CF* (*FCF*) which is a Future Particle attached to a Simple CF or to an RCF. These observations in fact imply a formal morphological grammar which we will refer to as *Morphology Structure rules*, or *MS rules*. We can define *MS rules* for rewriting Verb CFs in M.S.A. as follows:

a.i. FCF ⟶ (F) RCF

a.ii. RCF ⟶ PCF (CP)

a.iii. RCF ⟶ ICF (CP)

a.iv. PCF ⟶ P/R S1

a.v. ICF ⟶ P P/R S2

a.vi. P/R ⟶ pattern ... root ... pattern ... root

a.vii. F ⟶ {sa/li}

a.viii. CP ⟶ {niy/iy/ya/na|/ka/ki/kuma|/kumo/kunna/hu/hi/ha|/huma|/hima|/humo/himo/hunna/hinna}

a.ix. P ⟶ {:/t/y/n}

a.x. S1 ⟶ {otu/tu/ona|/na|/ota/ta/oti/ti/otuma|/tuma|/otumo/tumo/otunna/tunna/a/ay/ato/a|/ata|/uw|/awo|/ona/na}

a.xi. S2 ⟶ {u/a/o/uw/ay/uwna/uw|/awona/awo|/iyna/iy/a|ni/a|/ona/na}

As described in *a.*, rules *i.–xi.* are context-free rules for the *generation*, in the sense of formal description of valid morphological strings, that is, sequences of morphemes, and the description of *Morphology Trees*, or descriptors, which we call *M-Markers*. Looking at the rules in a top-down fashion we are allowed to construct an FCF by attaching an optional Future Particle F to an RCF using rule *a.i.* Rules *a.ii.* and *a.iii.* generate an RCF by attaching an optional Accusative Pronoun CP to a PCF or an ICF. Rule *a.iv.* rewrites a PCF as an obligatory lexical P/R attached with an obligatory suffix. Rule *a.v.* rewrites an ICF as an obligatory prefix, followed by an obligatory P/R combination attached with an obligatory suffix. Rule *a.vi.* rewrites a P/R combination as a discontinuous unit of pattern and root elements. Rules *a.vii.* to *a.xi.* are rules of *terminal* selection which allow the selection of affixes among finite sets of bound morphemes. For instance, Rule *a.viii.* allows the selection of affixes among the set listed in Table 21, while rules *a.ix.* to *a.xi.* allow the selection of perfect and imperfect affixes among the set of V-Affixes listed in Tables 9 and 10. Thus rules *a.i.* to *a.vi.* are rules for the generation of *non-terminal* symbols or morphological CATEGORIES in the grammar while rules *a.vii.* to *a.xi.* are rules for the selection of terminal CATEGORIES. The whole set of rules in *a.* describes a formal CFG for the generation of M-Markers of the type shown in Figures 4–5 (at the end of this Chapter).

II.5.2 Annotation of the Morphological Rules

The formal CFG described in II.5.1.*a.* above is too powerful or unconstrained to generate "all and only" the grammatical morphological sequences of Arabic. It overgenerates M-Markers by allowing invalid sequences of the type disallowed in § II.4.3,4 above. For instance, using rules *a.i.*, *a.ii.*, and *a.iv.* we are allowed to generate strings of the type '*sa$kataba', "*going to-he wrote". This was disallowed by rule II.4.4.*d.i.* Using the same rules we can generate '*li$yadorusuwna', "*in order for them to study". This was disallowed by rule II.4.4.*d.iii.* Further, the CFG in *a.* lacks specification of the Graphotactic Conditions (GCs) and allows sequences of the type '*qatalotumo$hu', "*you killed-him", thereby violating rule II.4.3.*a.*, '*svaribuw|$hu', "*they drank-it", thereby violating rule II.4.3.*b.*, and '*yah-omilu$hi', "*he carries-it", thereby violating the rule of Vocalic Compatibility (VCo) in II.4.3.*c.* We are also allowed to generate strings like '*kunona', "*they were", and '*waladna', "*they gave birth", using rule *a.iv.* and '*tad/onunona', "*they think", and '*tudarrisna', "*you teach", using rule *a.v.* These last CFs violate the graphotactic sequence rules of II.3.*a,b.* Rules *a.iv.* and *a.v.* also allow the generation of CFs such as '*banuw|', "*they built", '*jalasawo|', "*they sat", '*yuwladawo|', "*they are born", and '*yubonuw|', "*they are built". These CFs violate the Affix Selection Rules (ASRs) of II.3.*c,d.*

Therefore, we need a grammar that does not only have the power to generate all valid morphological sequences but also the power to constrain the generation to cover only valid

sequences. We then need a way of incorporating the different conditions into the CFG of II.5.1.*a*. We propose feature *annotations* that can be added to the rewrite rules to augment the CFG to a feature *Annotated*, or CSG, as in *a*. below:

a.i.	FCF	⟶	(F [+TENSE, +MDG]) RCF [+TENSE, +MOOD]
a.ii.	RCF	⟶	PCF [+GCs, +VCo] (CP [+GCs, +VCo])
a.iii.	RCF	⟶	ICF [+GCs, +VCo] (CP [+GCs, +VCo])
a.iv.	PCF	⟶	P/R [+ASRs] S1 [+ASRs]
a.v.	ICF	⟶	P P/R [+ASRs] S2 [+ASRs]
a.vi.	P/R	⟶	pattern ...root ...pattern ...root [+ASRs]
a.vii.1.	F	⟶	'sa' [+imperfect, +indicative]
a.vii.2.	F	⟶	'li' [+imperfect, +subjunctive]
a.viii.	CP	⟶	{niy/iy/ya/na|/ka/ki/kuma|/kumo/kunna/hu/hi/ha|/ huma|/hima|/humo/himo/hunna/hinna} [+GCs, +VCo]
a.ix.	P	⟶	{:/t/y/n}
a.x.	S1	⟶	{otu/tu/ona|/na|/ota/ta/oti/ti/otuma|/tuma|/otumo/ tumo/otunna/tunna/a/ay/ato/a|/ata|/uw|/awo|/ona/ na} [+ASRs]
a.xi.	S2	⟶	{u/a/o/uw/ay/uwna/uw|/awona/awo|/iyna/iy/a|ni/a|/ ona/na} [+ASRs]

Grammar II.5.2.*a*. differs from that in II.5.1.*a*. above in stating required conditions for different *rule contexts*. Thus, the Annotated Grammar is more constrained in generating only morphological sequences. We can read rule *a.i.* of this grammar as allowing a sequence F RCF $\mathcal{IFF}$ the feature values for TENSE and MOOD-GOVERNMENT requirements of F are equal to, or identical with, the TENSE and MOOD feature values of RCF. Further, the CATEGORY RCF has to satisfy the GCs of II.4.3.*a*,*b*. above according to rules *a.ii.* and *a.iii.* and to satisfy the VCo rule of II.4.3.*c*. according to the same rules. Rules *a.iv.* and *a.v.* force P/Rs and suffixes to be of compatible types and meet ASR requirements as specified in rules II.3.*c*,*d*. The specification of particular TENSE, MOOD, or other conditions is then spelled out by the specific lexical CATEGORY. Thus, in rule *a.vii.* the Particle 'sa' specifies an RCF of the TENSE and MOOD feature values [+imperfect, +indicative] and the Particle 'li' specifies an RCF of the TENSE and MOOD feature values [+imperfect, +subjunctive]. In rule *a.viii.* each lexical CP has to satisfy the said GCs and VCo to be able to attach with a PCF or an ICF. Rules *a.vi.*, *a.x.*, and *a.xi.* require that each lexical root and suffix specify their ASR type so as to allow the selection of appropriate suffixes for each root. Thus, the non-terminal symbols specify feature *labels* while lexical symbols specify feature *values*. The specific conditions, which operate as constraints on the output of the rewrite rules, are specified by individual lexical elements. Using this modified version of the grammar we can modify the M-Markers of

Figures 4–5 below to Annotated M-Trees of the type shown in Figures 6–7 and we can generate Verb sequences of the type in *b.* below:

b.i.	'rakiba', "he rode":	PCF
b.ii.	'rakiba$hu', "he rode it":	PCF$CP
b.iii.	'yarokabu', "he rides":	ICF
b.iv.	'yarokabu$hu', "he rides it":	ICF$CP
b.v.	'sa$yarokabu', "he is going to ride":	F$CF
b.vi.	'sa$yarokabu$hu', "he is going to ride it":	F$RCF
b.vii.	'li$yarokaba', "in order that he ride":	F$CF
b.viii.	'li$yarokaba$hu', "in order that he ride it":	F$RCF
b.ix.	'rakibuw|', "they rode":	PCF
b.x.	'banawo|', "they built":	PCF
b.xi.	'h-asunna', "they became beautiful":	PCF
b.xii.	'takunna', "they are":	ICF
b.xiii.	'yah-osibuw$hu', "they think-him":	ICF$CP
b.xiv.	'h-asibotumuw$hu', "you thought-him":	PCF$CP

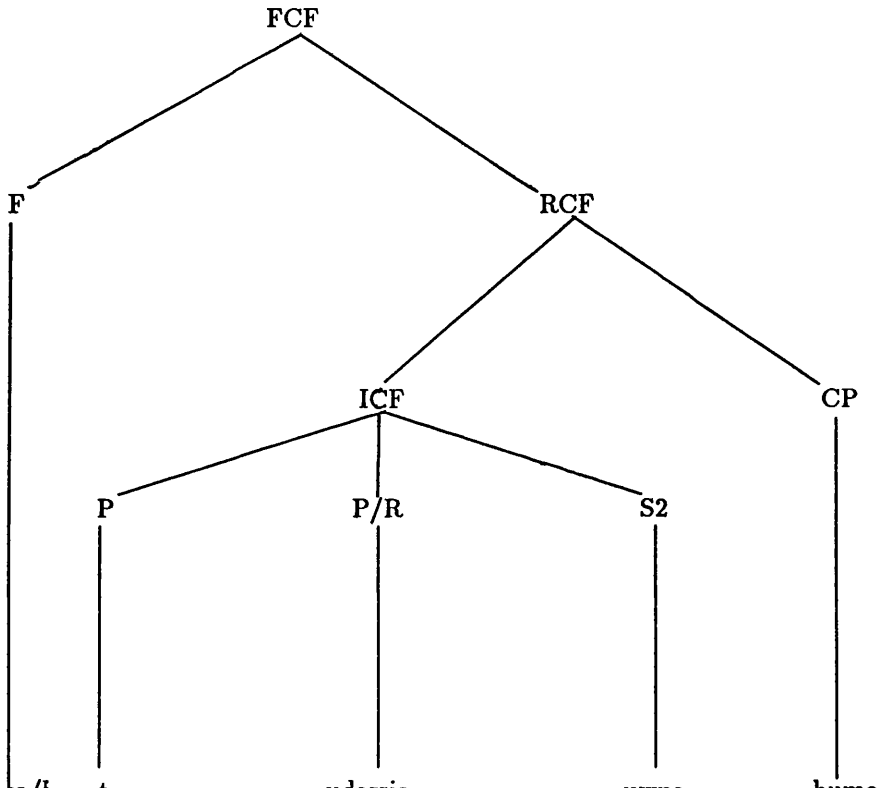


FIGURE 4: An M-Marker for a Complex CF of Type 1

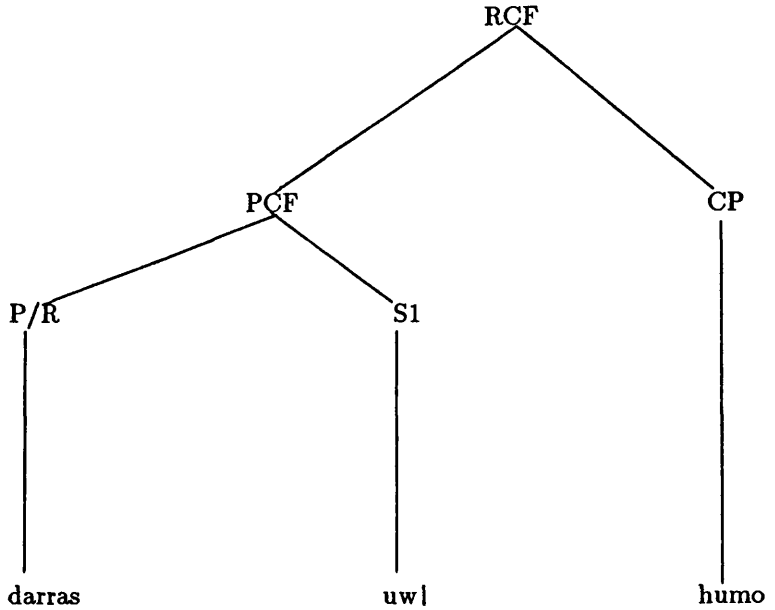


FIGURE 5: An M-Marker for a Complex CF of Type 2

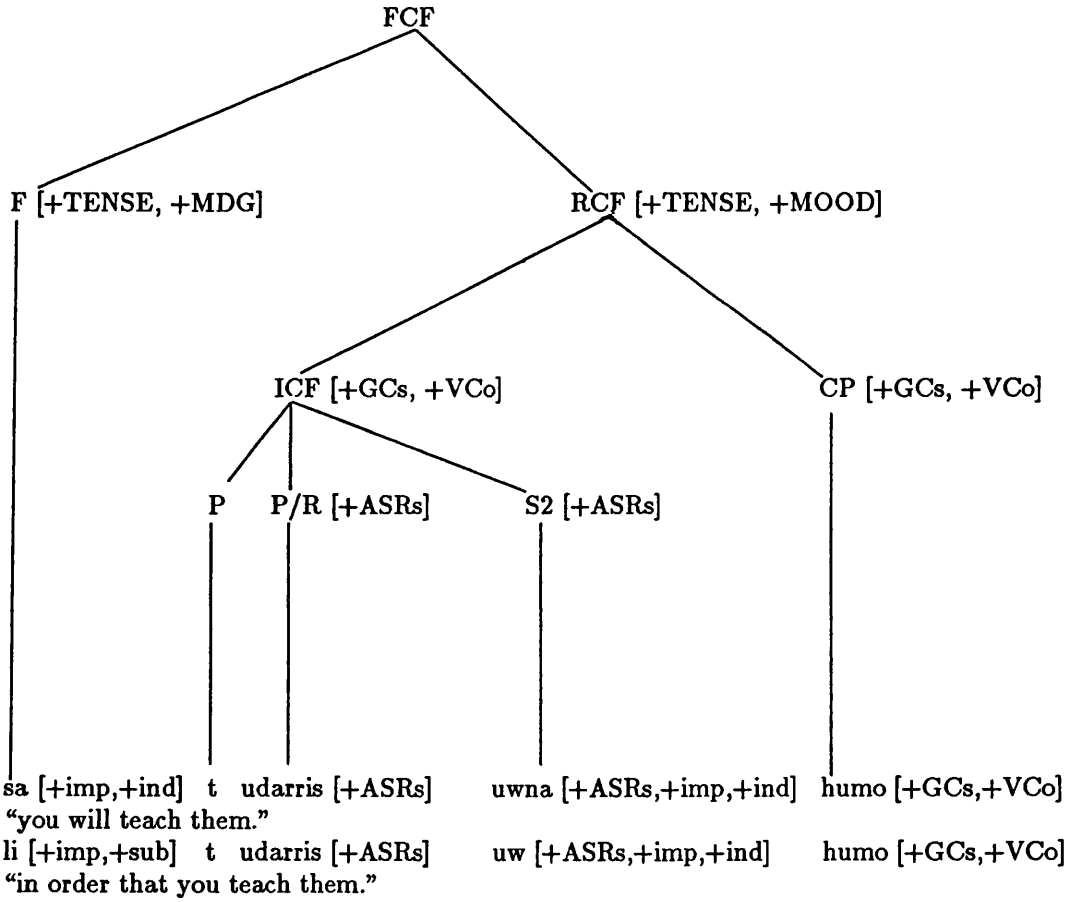


FIGURE 6: An Annotated M-Marker for a Complex CF of Type 1

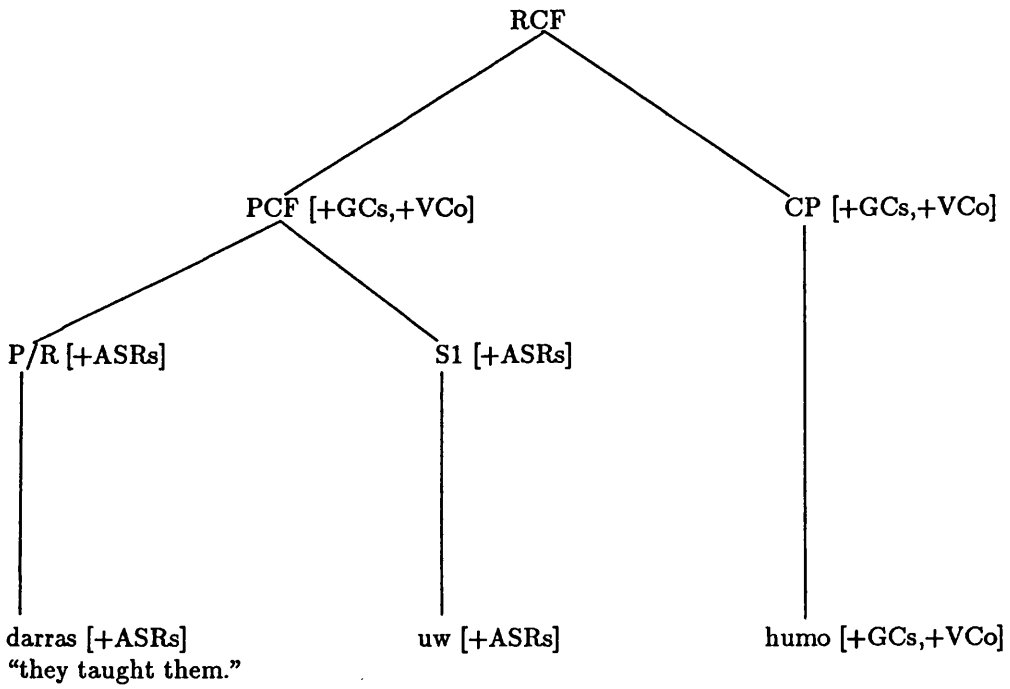


FIGURE 7: An Annotated M-Marker for a Complex CF of Type 2

==0== <*****> ==0==

Chapter 5

PROCESSING THE V-COMPLEX

INTRODUCTION

In Chapter 4, we provided structural descriptions for V-Complexes. In Chapter 5, we need to convert these descriptions to formal specifications which detail for each morphological structure a description, a basic structure, LENGTH and boundary conditions, and a role.

The structures thus defined will allow us to build in Section I a set of automatic dictionaries and property lists which we can use to construct a database for Verbs. The formal specifications reached will enable us to implement the Morphological Rules in a program that we can call the *V-Processor*. This processor needs to make use of the initial recursive closed subroutines defined in Chapter 3 with the aim of carrying out fast and primitive tasks such as SEGMENTation, word ASSEMBLY, MATCHing, and character PICKing. The V-Processor also needs to have intermediate recursive closed subroutines for root EXTRACTion and DERIVation, as well as higher-level recognition procedures. Such procedures have to be context-sensitive operations since they need to identify inputwords by identifying their constituent morphemes and detecting specific contexts that enable them to ASSIGN properties, disambiguate homonymic structures or raise ambiguity flags where resolution is not possible.

The V-Processor has to accomplish the tasks of V-Complex parsing but also to use devices like filters in order to ensure at the same time the monitoring of the linguistic conditions and constraints, and to use context identification in order to deal with the homonymy described in Chapter 4.

Section II of this Chapter will give a synthesis of the linguistic and computational structure of the V-Processor, describe the architecture and organization of the processing modules. It will also explain the search strategy on which the attempted recognition of an inputword is based. Examples of the performance of the V-Processor and samples of output and timings for V-Parsing tasks will be provided in Appendix D, § A.

I CONTENTS OF THE V-COMPONENT

I.1 THE V-DATABASE

I.1.1 Data Representation and Data Access

Before we tackle the processing of variable *template* structures (which we define as abstract composites of finite morpheme positions that are all potentially realizable but that may or may not be realized depending on the optionality or requirement of particular morphological categories), we need to look at computational data representation of constant lexical V-Structures. The importance of finding an adequate representation method for the data described in Chapter 4 and the discovery of canonical forms for these data, lies in the vital need for a viable and cost-efficient database where lexical structures can reside within records accessible to a given processor. Such a database needs to incorporate facilities for the storage of these data structures, ASSOCIATion between them and access to them by the processor.

The facilities provided in Cambridge LISP allow the definition and storing of several types of data structures as described in Chapter 3. Some of these system-defined facilities turn out to be completely appropriate for our purposes. The function SETQ, for example can be used to ASSIGN initial values to various data items. For instance, we classified roots into contrasting and sometimes overlapping groups such as causative versus intensive, regular versus irregular. We can use SETQ to write the statements in *a.*:

a.i. (SETQ causative '(ktb drs frsv ...)).

a.ii. (SETQ intensive '(fth- ksr qtl ...)).

a.iii. (SETQ regular '(ld rd h-d ...)).

a.iv. (SETQ irregular '(ls kn s;r ...)).

These function calls have the effect of creating four *dictionaries*, namely lists of lexical entries with the first arguments being the identifiers, or keys, and the second arguments being the actual lexical entries ASSOCIATed with the dictionary names. The function call:

b. (FLUID '(causative intensive regular irregular))

ensures that these dictionaries now have dynamic scope and can thus be directly accessible or manipulated. For any one of these dictionaries we can pinpoint positions in it as desired and request specific elements of it, we can retrieve the whole list or parts of the list; DELETE, SUBSTITUTE or update parts of it; APPEND to it; COPY it; verify constituent MEMBERship; compare it with other lists; as well as check its current LENGTH, status, and value, all the time using system-defined functions such as those in Chapter 3, § I.4.

The definition of such dictionaries as causative, intensive, and so on allows the constraining of database access if so desired. Specifically, particular rule applications can be restricted to specific dictionaries. Such rules will be said to be *dictionary-dependent* rules. For instance, rules of TRANSITIVITY PROMOTion can be applied to causative dictionaries only.

However, we may sometimes need even more complex types of data structures than the above. This turns out to be the case for lexical entries such as roots and patterns and the problem is that of finding a method for binding a given pattern or patterns to a given root in such a way as to ensure a two-way ASSOCIATion between them and to exclude inappropriate patterns from being related to a given root. The solution adopted was to use association lists in order to bind particular patterns with particular roots. But how can we create these association lists in the first place? Simple SETQ ASSIGNment is too complex for building association lists: it requires two ASSIGNments, one to SET up roots and one to SET up patterns, in addition to requiring some complex formula for ASSOCIATion between them. The simplest solution was to take advantage of the PLIST structure as an association list. It has the advantage of SITting up patterns as property values for radical entries and therefore each pattern will be bound to the correct root and the correct root only. The only modification that we have to introduce is to use lists and not identifiers as property values.

The use of lists as property values is allowed in CL and it has the added advantage of allowing the storage of more than one pattern just in case a given root requires more than one. This is exactly the case for irregular roots, for which we noted a general distinction between third-PERSON and other CFs. To take the example of 'kn', "to be", we can SET up a root-pattern ASSOCIATion structure as follows:

c. (PUT pattern 'kn '(cuc ca|c acuwc acuc)).

The root-pattern PLIST called simply a *PATLIST*, is then created as a result of using the function PUT. To introduce further precision in statement *c.* above we can now implement the classification introduced in Chapter 4 for pattern types as a distinction of PLISTS in order to anticipate eventual DERIVation from these entries. Namely, we can build four types of PATLIST:

d.i. Perfect active.

d.ii. Perfect passive.

d.iii. Imperfect active.

d.iv. Imperfect passive.

The LISP expression in *c.* above can now be reformulated as two statements:

e.i. (PUT 'perfectpattern 'kn '(cuc ca|c)).

e.ii. (PUT 'imperfectpattern 'kn '(acuwc acuc)).

Such distinctions allow the constraining of database access if so desired. Indeed, we now have to turn our attention to such access. The retrieval of particular patterns from the appropriate PLIST can now be done in a simple and direct manner. The function ATSOC allows free access to the patterns using the expression:

f. (ATSOC root (PLIST 'patterntype)).

This call will RETURN simply and exclusively the patterns that are in the pattern-type PLIST of a given root.

I.1.2 Property ASSIGNment

With each lexical entry of the V-Database is ASSOCIATed a number of FEATURES. Each feature has a *feature label* which is its name, and a *feature value* which is the value of the given entry for that feature. In parsing, feature ASSIGNment is a complex issue and particularly so for Arabic. This is because of the multiplicity of features and types of ASSIGNment on the one hand and the homonymy problems outlined in Chapter 4, § II.2, on the other. Essentially, we described two types of properties or features:

- UNIQUE properties.
- HOMONYMIC properties.

The ASSIGNment of unique properties in CL is a straightforward one using the usual property mapping functions PUT and MAPC. The retrieval of the properties thus ASSIGNed is normally achieved by invoking GET. We can augment the speed of such access by using MACROS (as defined in Ch. 3, § I.2), to carry out these tasks. The general form of such macros is:

(DM *MACRO-NAME* (identifier) '(GET '*propertyname* ,identifier)).

However, the ASSIGNment of homonymic properties presents us with a problem. We distinguished in Chapter 4, § II.2, two types of homonymy, one that is resolvable in the morphological context, i.e., at each level of concatenation, and one that is not resolvable. The answer is to

make use of the property labels and the property values in the way we describe immediately below. First, we note that the ASSIGNment of all properties, in the V-Database, follows the descriptions of Verb properties as specified in Chapter 4. This ASSIGNment is then made as follows:

a. ASSIGN *unique* values to a lexical entry initially as *default* values. Then ASSIGN or MODIFY these initial values at run time as *fixed* or as *dynamic* values. For instance, the suffix 'ona' initially carries in the database the value *third* for the property name *PERSON*. Given the environments:

- 'ona' / [ ] — perfect P/R,
- 'ona' / [ ] — <y>$imperfect P/R,

the value *third PERSON* is ASSIGNed by the program as fixed value. Given the environment:

'ona' / [ ] — <t>$imperfect P/R,

the value *second PERSON* is now ASSIGNed as a fixed value again. Some values are context independent, such as the prefix <:> which always has the values: {first, singular} for the property labels *PERSON* and *NUMBER* respectively. Here is a breakdown of feature labels followed by their unique values as defined in the V-Processor.

NUMBER:	singular, dual, plural.
GENDER:	masculine, feminine.
PERSON:	first, second, third.
MOOD:	indicative, subjunctive, jussive.
CATEGORY:	VB, VR, PVB, PVR.
TENSE:	perfect, imperfect.
VOICE:	active, passive.
REFERENCE:	exlocutive, collocutive.
MOOD GOVERNMENT:	indicative, subjunctive, jussive.
TRANSITIVITY:	intransitive, ptransitive, monotransitive, ditransitive, cotransitive, xtransitive, ptransitive1, ptransitive2, nctransitive.

b. ASSIGN *neutral* values to lexical entries that have potentially resolvable homonymy and to entries that will always have neutral values. In this thesis, we use the term *neutral* to characterize a common form which is used in Arabic for the same property, such as NUMBER or GENDER, but which has two or more values for that property. For instance, the prefix <:> has the property values *masculine* and *feminine* for the property *GENDER* in all contexts. In this sense, <:> will always stay neutral for GENDER. To handle this particular kind of

homonymy, we used neutral, or *integrated* labels such as *mf* for masculine-feminine. Disambiguation when possible is context-dependent and subject to concatenation, and the principle behind the ASSIGNment of neutral values to homonymic objects is that these values can either be disambiguated inside the morphological component, or, if this is not possible, they can offer flexible choices for the eventual syntactic analysis. For instance, an entry with neutral GENDER could be made to agree with another entry that is neutral in GENDER, or that is masculine or feminine. The Verb 'takotubu', "you write", with neutral PERSON and GENDER agrees with ':anota', "you", 2 (M) S and with ':anoti', "you", 3 (F) S. Here is a breakdown of feature labels followed by their neutral values and integrated names together with some examples as defined in the V-Processor:

- NUMBER:
  - singular-plural: **sp**, e.g., <t>$passive Defective imperfect P/R$'ona', such as 'tubonayona', "you are built".
  - dual-plural: **dp**, e.g., the prefix <n> in all contexts, 'nah-onu', "we".
- GENDER: masculine-feminine: **mf**, e.g., the prefix <:>.
- PERSON: second-third: **rx**, e.g., <t>$imperfect P/R$<u>.
- MOOD:
  - indicative-subjunctive-jussive: **jis**, e.g., <t>$active imperfect P/R$'ona'.
  - subjunctive-jussive: **sjj**, e.g., <t>$imperfect P/R$'uw|'.
  - indicative-subjunctive: **ids**, e.g., <t>$imperfect Defective P/R$'iy'.
- CATEGORY:
  - Verb-Noun: **VN**, e.g., 'ɗ-ahaba', "he went/gold".
  - Verb-Adjective: **VA**, e.g., ':axod;aru', "I become green/green".
  - Verb-Numeral: **VM**, e.g., 'ɛasvara', "he joined a group of ten/ten".
  - Referential Verb-Annexed Noun: **WV**, e.g., 'jabalahu', "he created him/his mountain".
- TRANSITIVITY: intransitive-cotransitive-nctransitive: **nctr**, e.g., 'h-usiba', "it was thought/it was counted".

c. ASSIGN *ambiguous* values to lexical entries that have homonymic values unlikely to be resolved by the morphological analyser. The distinction between the neutral and ambiguous property values is that both are potentially resolvable but the latter less likely so than the former, but if the first remains after MA, the syntactic component can simply treat it as neutral,

i.e., offering a flexible choice of agreement possibilities as described above; if ambiguous values are left, these can be seen as an initial task for the syntax to solve before parsing can be resumed. There is in fact a small residue of such ambiguity which remains at the end of MA, which is given values that we can look at as ambiguity *flags* to be picked up by the syntactic component. Such flags were marked in the program with a prefix *x* for unknown. To give an example, in fact the only such example in the V-Processor of morphologically unresolvable ambiguity, the configuration: <t>$active Defective imperfect P/R$'iy' is concurrently ambiguous for the property *MOOD* between the values: indicative and subjunctive and for the property *PERSON* between the values: *second* and *third*. We can envisage for this example the following syntactic environments:

- ':anota taboniy', "you build": 2 (M) S, indicative.
- 'hiya taboniy', "she builds": 3 (F) S, indicative.
- ':anoti lano taboniy', "you will not build": 2 (F) S, subjunctive.
- ':anoti lamo taboniy', "you did not build": 2 (F) S, jussive.

In this case the flag *xds* is used to signal both NUMBER and MOOD ambiguities.

d. ASSIGN two or more labels in the case of lexical entries that have more than one value for a given feature. For instance, in the case of radical homonymy described in Chapter 4, § II.2.5, both the roots 'h-sb', "to think/count" and 't-mn', "to join a group of seven/become expensive", have more than one pattern for the perfect and for the imperfect TENSE depending on which meaning is chosen. For instance, 'h-sb' has the perfect patterns 'cacic' for "thought" and 'cacac' for "counted". If we proceed as usual, viz.,

- (PUT 'perfectpattern 'h-sb '(cacic)),
- (PUT 'perfectpattern 'h-sb '(cacac)).

The second statement will just erase the effect of the first one by replacing the value ASSIGNed in it by the second value. This is because any identifier such as the property name: *perfectpattern* is treated as a unique key by the LISP interpreter for a PLIST with that name. Since we want to preserve both values the solution we suggest is to use two different names for the *perfectpattern* label of homonymic roots, thus getting the LISP interpreter to treat them as if they were two different properties but use the two PLISTS thus created as if they were only one list for perfect patterns. We can then reformulate the above statements as follows:

- (PUT 'perfectpattern1 'h-sb '(cacic)),
- (PUT 'perfectpattern2 'h-sb '(cacac)).

The unification of the treatment of the two PLISTS thus created is made at program rather than database level.

I.1.3 The Structure of the V-Database

Having defined ways of handling data representation and access, and property ASSIGNment and retrieval for V-Data structures, we can now formally define the *V-Database* and specify its contents. The *V-Database* is then simply a superset which has dual structure consisting of a set of dictionaries defined by direct ASSIGNment using the function SETQ in the case of LISP, therefore initializing the set of dictionaries as FLUID VARIABLES with dynamic unrestricted scope to their given values. We hinted that the specification of particular dictionaries can then be used to constrain access if such restriction is desired.

The second constituent part of the V-Database is a set of properties and property values including feature values and association lists and defined on the lexical entries by indirect ASSIGNment using the functions PUT and MAPC therefore initializing the set of lexical entries to a number of property values that may initially be default and contextually fixed or dynamic. Homonymic properties are neutralized if at the end of MA they are unresolvable and carry ambiguity flags in anticipation of possible disambiguation at the syntactic level. Fixed properties are explicitly unique.

Earlier, we defined a dictionary as a set of lexical entries with an identifier as the dictionary name and a value which is the constituent MEMBERship of the set. We can add that each of these entries is ASSOCIATed with one or more PLISTS with given feature labels that carry its default property values. The set of dictionaries and properties thus created constitute the contents of the Verb database in TUNIS1. These contents (of which a complete description is given in Volume II, Appendix B), can be summarized as follows:

a. A dictionary for V-Roots: the set of all the radical entries carrying the default CATEGORY: VB and initial TRANSITIVITY values: {ntr, ptr1, ptr2, ctr, mtr, xtr}. Augmented roots carry only CATEGORY-feature values, their TRANSITIVITY values being detected at run time, i.e., during the progress of a given parse.

b. A dictionary of pattern ASSIGNments: the set of roots each being ASSOCIATed with its characteristic patterns. The pattern lists thus created carry feature values for TENSE: perfect-imperfect and VOICE: active-passive.

c. A set of affix dictionaries as follows:

c.i. A dictionary of Sound perfect V-Suffixes.

c.ii. A dictionary of Defective perfect V-Suffixes.

c.iii. A dictionary of imperfect V-Prefixes.

c.iv. A dictionary of Sound imperfect V-Suffixes.

c.v. A dictionary of Defective imperfect V-Suffixes.

Each of these affixes carries its own default property values for NGP and MOOD.

c.vi. A dictionary of bound enclitic Pronouns with NGP, CATEGORY, and REFERENCE features and feature values.

c.vii. A dictionary of Future Particles with their subjunctive and indicative lexical specification for MOOD GOVERNMENT.

c.viii. A dictionary of V-PATLISTS.

d. Additionally there is a dictionary for Subject Pronouns with their NGP values and other values that are of relevance to the Nominal, or N-Processor.

From this description, we can see that the distribution of properties is not only defined on lexical entries such as affixes, roots, and patterns but also on identifiers such as PATLISTS. These *superproperties* in the sense of being properties of property lists, can in fact be used to ASSIGN TENSE and VOICE for instance, but also to distinguish patterns just in case of ambiguity such as described in Chapter 4, § II.2.1 and § II.2.2, by using PATLIST MEMBERship. For the purposes of this exposition, Appendix B lists each lexical entry in a uniform way together with its ASSOCIATed PLISTS on a new line. The PLISTS can be either PATLISTS or feature PLISTS (which are SET up in a collective list called *PROPLIST1* that includes all the properties defined in Chapter 1, § II.3.1). Appendix B lists PATLIST entries in this left-to-right order:

- entries in active perfect PATLIST1 and in passive perfect PATLIST1.
- entries in active imperfect PATLIST1 and in passive imperfect PATLIST1.
- entries in active perfect PATLIST2.
- entries in active imperfect PATLIST2.

These lists are merged into two *Global Pattern Lists* or supersets called "*PATLISTP*" for the perfect pattern lists and "*PATLISTI*" for the imperfect pattern lists. In addition to the dictionaries, there are other forms of lexical entry that serve the purposes of classification rather than ASSIGNment. Among these are the dictionaries mentioned in § .1.1 above and which are:

e. A causative dictionary containing a set of augmentable roots with *upwardly mobile* TRANSITIVITY values.

f. An intensive dictionary which is a set of augmentable roots with *fixed* TRANSITIVITY values.

g. A regular dictionary made up of a set of subdictionaries for Sound and Defective roots that have regular conjugation.

h. An irregular dictionary made up of a set of subdictionaries for Sound and Defective roots that have irregular conjugation.

These classifications are motivated by the need for dictionary-dependent rules for the purposes of accurate DERIVations and TRANSITIVITY mapping. Besides classification dictionaries, there are other dictionaries of a general type and which are listed in Appendix B, as follows:

- A *lexicon* containing the set of all lexical entries in TUNIS1 and supplied for information.
- A *rootlist* which is the set of all 149 roots used by the program for the DERIVation of Verbs, Verbals, and non-Verbal Nominals.
- The list of all patterns supplied for information. The pattern lists used by the V-Processor reside in the PLIST structures.
- A list of error-messages for errors that are detectable by TUNIS1. These are coupled with potential erroneous input to form a more complete message.

I.2 THE V-PROCESSOR

I.2.1 Initial Closed Subroutine: Root EXTRACTion

In Chapter 4 we provided a description of V-Roots which we can use to build a formal specification for a V-Root as in Table 25 below.

In addition to Table 25, we need a basic subroutine that is able to use the above specification in order to identify roots. This subroutine will take two arguments, an inputword and a rootlist as provided in the V-Database. Given the inputword which here is a Conjugation Form, the routine needs to be able to extract from it a non-vocalic form which is the set of consonants and semivowels and then *prune* it by performing predetermined simple linguistic transformations on it in order to obtain the true *radical* form, i.e., the actual lexical root that is to be checked against the rootlist, then it has to ASSIGN to it the appropriate TRANSITIVITY value.

Thus, the routine, call it *EXROOT*, for root *EXTRACT*ion, needs to combine linguistic and programming operations. The nature of the programming operations is not recursive as in

DESCRIPTION TYPE	a surface lexical item.
STRUCTURE	*BASIC STRUCTURE:* a canonical sequence of consonantal radicals. *CONCATENATION ORDER:* as given, left-to-right. *ROOT TYPES:* (a) *Basic:* i. *Sound:* with complete consonantal contents. ii. *Defective:* with abstraction of semivocalic contents. (b) *Augmented:* i. *Sound:* with infixation of extra radicals. ii. *Defective:* with abstraction of semivocalic contents and infixation of extra radicals. *EXAMPLES:* (a) i. ktb ii. kn. (b) i. kttb ii. gnn.
CONSTRAINTS	*GRAPHOTACTIC CONDITIONS:* none. *LENGTH:* L = [2, 4]. *BOUNDARIES:* unknown.
ROLE	constrain lexically the realization of CFs in conjunction with V-Patterns.

TABLE 25: A Formal Specification for a V-Root

SEGMENTation but procedural: it is subject to the nature of linguistic rules of this type: *if* case *n then* apply rule *x*. However, using the LENGTH constraints specified in Table 25 the routine has to FAIL if these constraints are violated. Nevertheless, note that Scheme 7 (in App. C) has modified values for LENGTH which also accommodate the *VERBAL* (traditionally called *PARTICIPLE* and defined in Chapter 6) root. These are STEPS IV, VII and VIII and will be explained in Chapter 7. The linguistic rules applied in EXROOT distinguish purely consonantal forms from consonantal and semivocalic forms since Augmented forms of Defective Hollow roots behave not like irregular but like regular roots. The rules also distinguish intensive and causative roots which determine the ASSIGNment of appropriate TRANSITIVITY values to roots. The simple transformations are able to prune non-vocalic forms by:

a. DELETing Augmentative infixes in order to obtain a basic radical form.

b. DELETing elements which we will call *Supernumerary* characters. We mean by *Supernumerary* characters those which occur besides radical elements in a morphological form without altering the meaning of the root, and which do not carry any linguistic information or properties. These are the set: {:, t, m}.

c. DELETing semivowels from non-Hollow Defective roots in order to obtain a MATCH with their canonical form in the database.

From a programming point of view the rules are organized in an order which means the longest forms get tackled first and the shortest ones last. This ensures that the *IF* conditions imposed are not mutually exclusive. For instance, if a character happens to be in the Supernumerary characters set, it may in fact be part and parcel of the root proper. In that case we would not wish to DELETE it, a move which would *maim* the root and render it unrecognizable. The root 'mnɛ', "to prevent", for example, has an initial <m> which is homonymic with a Supernumerary <m>. To avoid DELETing this <m> and generating the non-root '*nc', the conditions on LENGTH place such roots at the top before DELETions can take place. Thus the forms with a minimal LENGTH are directly checked against the database, then the longest forms undergo simple transformational operations. This means in fact keeping the LENGTH of a canonical form to the specified LENGTH interval: [2, 4]. Figure 8 below highlights the main functional flow of EXROOT.

I.2.2 Intermediate Closed Subroutine: DERIVation

Having defined an algorithm for V-Root EXTRACTion, we can now define an intermediate subroutine that uses that algorithm on an inputword in order to obtain a regular or irregular Verb *DERIVation*. Earlier, we defined regular Verbs as those having a unified pattern distribution for all PERSONS and irregular ones as having a different pattern distribution for different PERSONS, generally one pattern form for the third PERSON and another for the first and second. The details are specified in Tables 11 and 12.

Thus, a *V-DERIVation* is obtained by first EXTRACTing a root from the inputword and then using a specified *Global Pattern List* (GPL) which is itself a set of association lists in order to retrieve *Local Pattern Lists* (LPLs) which are PAIRings of roots and patterns. In § I.1.3, we defined two GPLs: "PATLISTP" and "PATLISTI" for perfect and imperfect pattern lists respectively. The specification of one list or the other is passed down from a subprogram that is higher than DERIVation. Each LPL belonging to the GPL is then scanned to retrieve one pattern that MATCHes the inputword. When one is found the DERIVation is successful and RETURNS the name of the LPL, the root and the pattern. These will be useful in imposing constraints on V-Conjugations, ASSIGNing particular properties and disambiguating homonymic structures. However, since we wish to guide the DERIVation so as not to allow the DERIVation of third-PERSON forms from a pattern restricted to first and second PERSONS, we cannot define *DERIVE* as a unified operation but rather as three different operations: *DERIVE1* DERIVES irregular third-PERSON types by EXTRACTing the root from the list of irregular roots and always picking the second pattern in the LPL; *DERIVE2* DERIVES

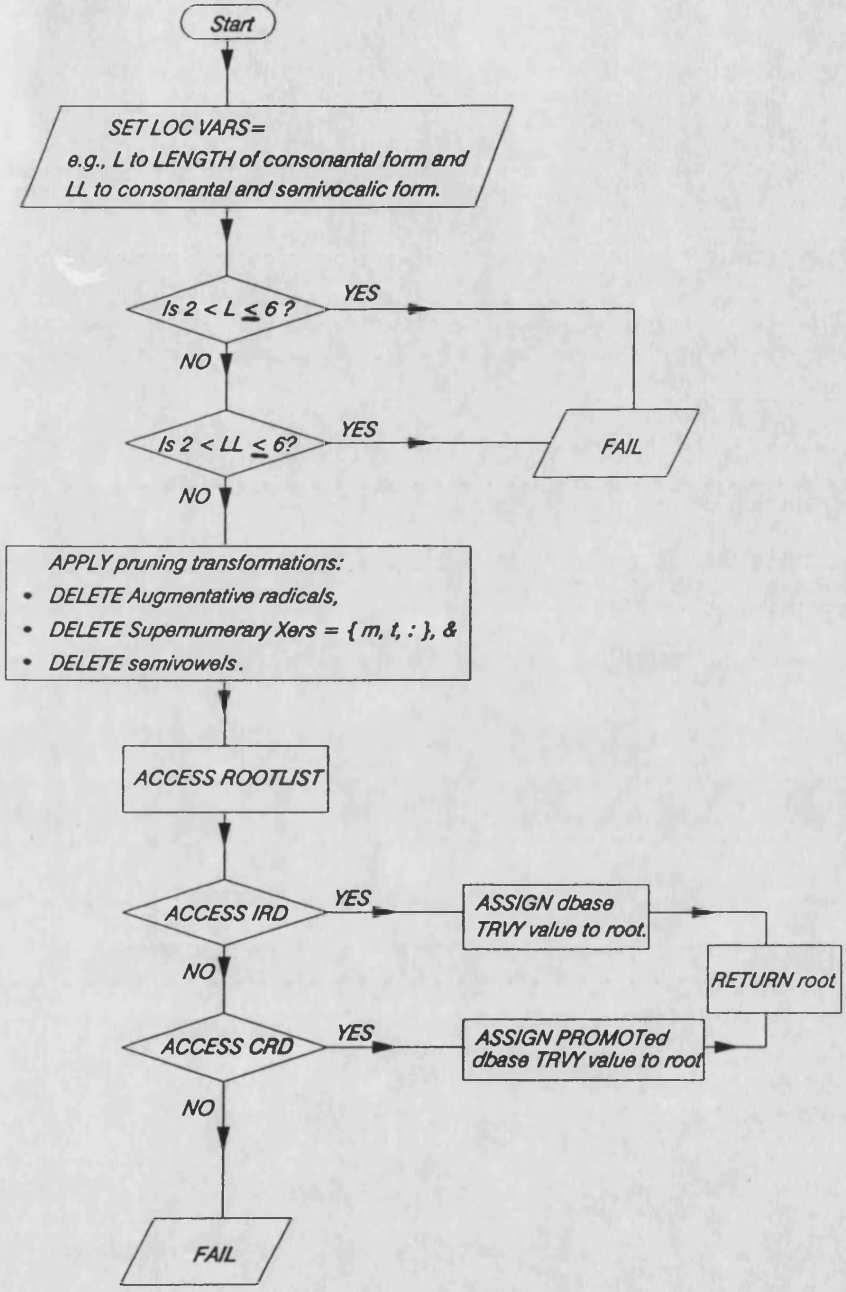


FIGURE 8: A Representation of the Root EXTRACTion Subroutine

irregular first- and second-PERSON types by EXTRACTing the root from the list of irregular roots and always picking the first pattern in the LPL; *DERIVE3* is slightly more complex but is designed to DERIVE regular types for all PERSONS. This is done by EXTRACTing a root which belongs to the list of regular roots and always picking the first pattern in the LPL. However, there is another group of roots, namely the Augmented Defective Hollow roots, which although their basic counterparts have irregular conjugation, have regular conjugation themselves as explained in Chapter 4, § II.1.2. For this group, DERIVE3 EXTRACTS the root from the irregular list but picks up the first pattern in the LPL. This will then enable the program that calls DERIVation to make a distinction between Sound and Defective roots that have a regular conjugation by calling DERIVE3, and those that have irregular conjugation by calling DERIVE2, for the third PERSON, and DERIVE1, for the first and second PERSON. Figure 9 below summarizes DERIVE1, DERIVE2, and DERIVE3 as the generalized DERIVation algorithm: *DERIVE*.

I.2.3 The Main V-Recognition Procedures

I.2.3.1 The Perfect CF Recognition Procedure

In Table 26 below, we provide a detailed specification for Perfect CFs as described in Chapter 4.

Now that we have defined initial and intermediate subroutines to EXTRACT roots from inputwords and DERIVE Verbs using SEGMENTation and MATCHing, we are in a position to define procedures for the recognition of CFs. We start with the definition of a procedure that uses the specification of Table 26 for the recognition of Perfect CFs, which we shall call *MKPERFECT*.

Given an inputword v for Verb, MKPERFECT needs to be able to recognize, if it is a Simple Perfect CF by SEGMENTing it into a centre and a suffix, conjugating the centre in the perfect active or passive form, by obtaining a *valid* DERIVation for it, viz., a valid lexical root and a MATCHed pattern ASSOCIATed with the root, and by identifying the suffix among the perfect V-Suffixes in the database. During the parsing progress, MKPERFECT needs to *PASS ON* to v the default NGP values of the suffix, the VOICE value of the LPL as ASSIGNed in the database and the CATEGORY and TRANSITIVITY values of the root as ASSIGNed in EXROOT. MKPERFECT also has to ASSIGN to v a TENSE value: *perfect*. Finally it needs to RETURN a result which is a list of the centre, the suffix, the name of the LPL, the root, and the pattern. However, both the DERIVation of the centre and the ASSIGNment of property values are context-sensitive operations. DERIVation is made using the GPL: "PATLISTP" which is a set of LPLs including roots ASSOCIATed with patterns. There are three broad categories of DERIVation covered by DERIVE1, DERIVE2, and DERIVE3 as described above.

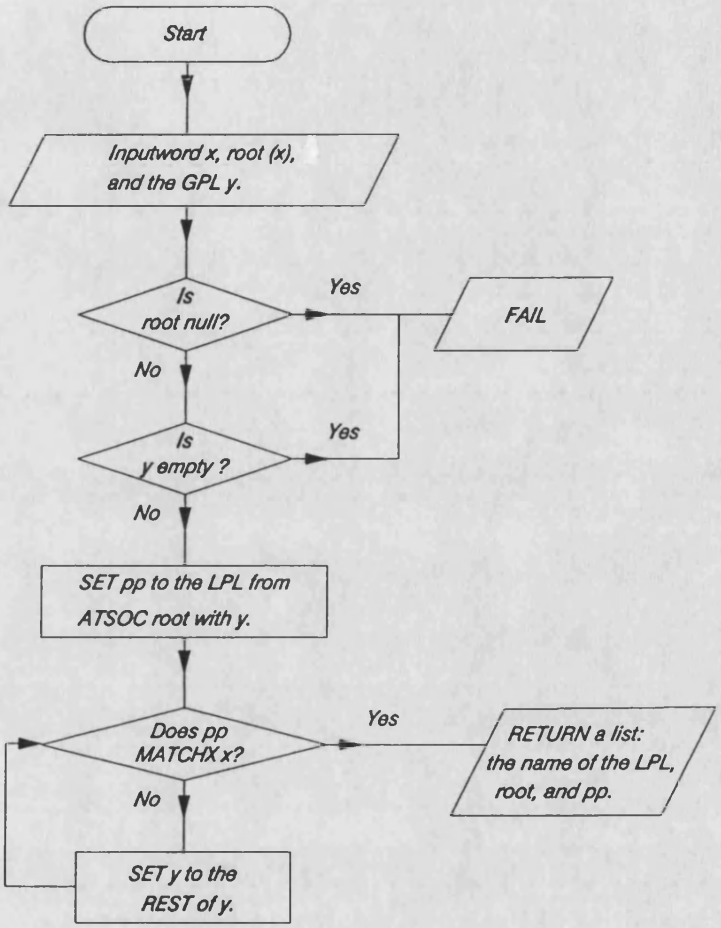


FIGURE 9: A Representation of the Verb DERIVation Subroutine

DESCRIPTION TYPE	an abstract composite template structure.
STRUCTURE	*BASIC STRUCTURE:* a sequence of an obligatory P/R combination slot and an obligatory suffix slot. *CONCATENATION ORDER:* given as P/R + S. *CF TYPES:* (a) *Sound.* (b) *Defective.* *EXAMPLES:* (a) cacac$S. (b) ca\|c$S.
CONSTRAINTS	*GRAPHOTACTIC CONDITIONS:* include boundary conditions such as Affix Selection Rules. *CF LENGTH:* unknown but ≥ 4. *P/R LENGTH:* the specified LENGTH. *S LENGTH:* L = [1, 6]. *CF BOUNDARIES:* <1 (CF) = unknown; and >1 (CF) = >1 (S). *S BOUNDARIES:* >1 (S) = {a/i/o/u/y/|}. *P/R BOUNDARIES:* the defined boundaries. *LEXICAL SPECIFICATIONS:* (a) GCs (S) = GCs (P/R); and (b) ASRs (S) = ASRs (P/R).
ROLE	conjugate Verbs with all SPs in the perfect TENSE using P/Rs and suffixes.
SIDE EFFECTS	possible NGP disambiguation.

TABLE 26: A Formal Specification for a Perfect CF

Having obtained a valid DERIVation, MKPERFECT then needs to exclude certain Defective roots from having non-valid suffix allomorphs as specified in Chapter 4, § II.3. For instance 'awo|' is the valid 3 (M) P suffix allomorph for the root 'bn', "to build" conjugated in the perfect active but the 3 (M) P allomorph 'uw|' is the valid suffix for that root conjugated in the perfect passive. Hence, MKPERFECT needs to distinguish VOICE and types of roots.

Another problem faced by MKPERFECT is that certain Defective Verbs cannot have the allomorph <a> for 3 (M) S but have to have one of these two: {ay, a|}. These Verbs also need to be distinguished on the basis of root and VOICE as specified in Chapter 4, § II.1.2.1. The ASSIGNment of property values is achieved on a contextual discrimination basis in one of two ways: either default values are inherited or new values are ASSIGNed at run time. Inheritance is allowed if the property values are unique and the context is appropriate or if the values are homonymic but they cannot be disambiguated at this stage. The function *PASSONPROP*, or

PASSON for short, has the task of ASSIGNing the default values of a smaller unit, such as a suffix, to a larger unit, such as a CF. We can assume *PASSON* to have the general form:

(PUT '*propertyname* 'identifier$_2$ (GET '*propertyname* 'identifier$_1$)).

To give an example of property ASSIGNment the suffix 'a|' carries the NGP values: RX (D) MF. However, for the perfect regular Sound Verbs the value *RX* has to be MODIFIed to *third*.

Further, MKPERFECT needs more details in order not to search blindly every inputword to see if it is a perfect Verb. Here, we can use the specifications in Table 26 to direct the parsing. For instance, using the right-hand boundaries of a Perfect CF, an ENTRY condition can restrict the operation of the MKPERFECT procedure to inputwords that conform to the condition. During implementation, this constraint was found to reduce parsing time to less than half the time taken without the entry condition imposed. Another constraint that is useful is the suffix LENGTH. Using the interval [1, 6], MKPERFECT can specify consecutive indexes for SEGMENTation such that an iteration is performed for that interval, and each time a new index value is specified and a SEGMENTation obtained, the left-to-right process of DERIVation for centre and identification for suffix is attempted until a valid DERIVation-suffix configuration is obtained, the *property heritage* which is the set of inherited values for particular properties by a larger unit from a smaller one is ASSIGNed and the result RETURNed. The procedure starts at the threshold 1 and keeps incrementing it by 1 and when the upperbound 6 is exceeded the parse is aborted. Thus, MKPERFECT is a left-to-right context-sensitive iteration for the interval [1, 6]. The logical structure of MKPERFECT is outlined in Figure 10 below.

I.2.3.2 The Imperfect CF Recognition Procedure

From the description of CFs in Chapter 4 we can deduce a formal specification for Imperfect CFs as in Table 27 below.

Now, using the SEGMENTation and DERIVation subroutines in a parallel way to the MKPERFECT procedure, and the specification of Table 27, we can build another procedure for the recognition of Imperfect CFs that we will call *MKIMPERFECT*.

Given an inputword v for Verb, MKIMPERFECT needs to be able to recognize if it is a Simple Imperfect CF by SEGMENTing it into three main component segments: a prefix, a suffix, and a centre; obtaining a valid DERIVation for the centre, to wit, a lexical root and a MATCHed pattern ASSOCIATed with the root, and successively identifying the suffix and the prefix among the imperfect affixes in the database, thereby conjugating the centre in the imperfect active or passive form. During the parsing progress MKIMPERFECT has to PASSON to v the default NGP and MOOD values of the affixes, the VOICE value of the LPL

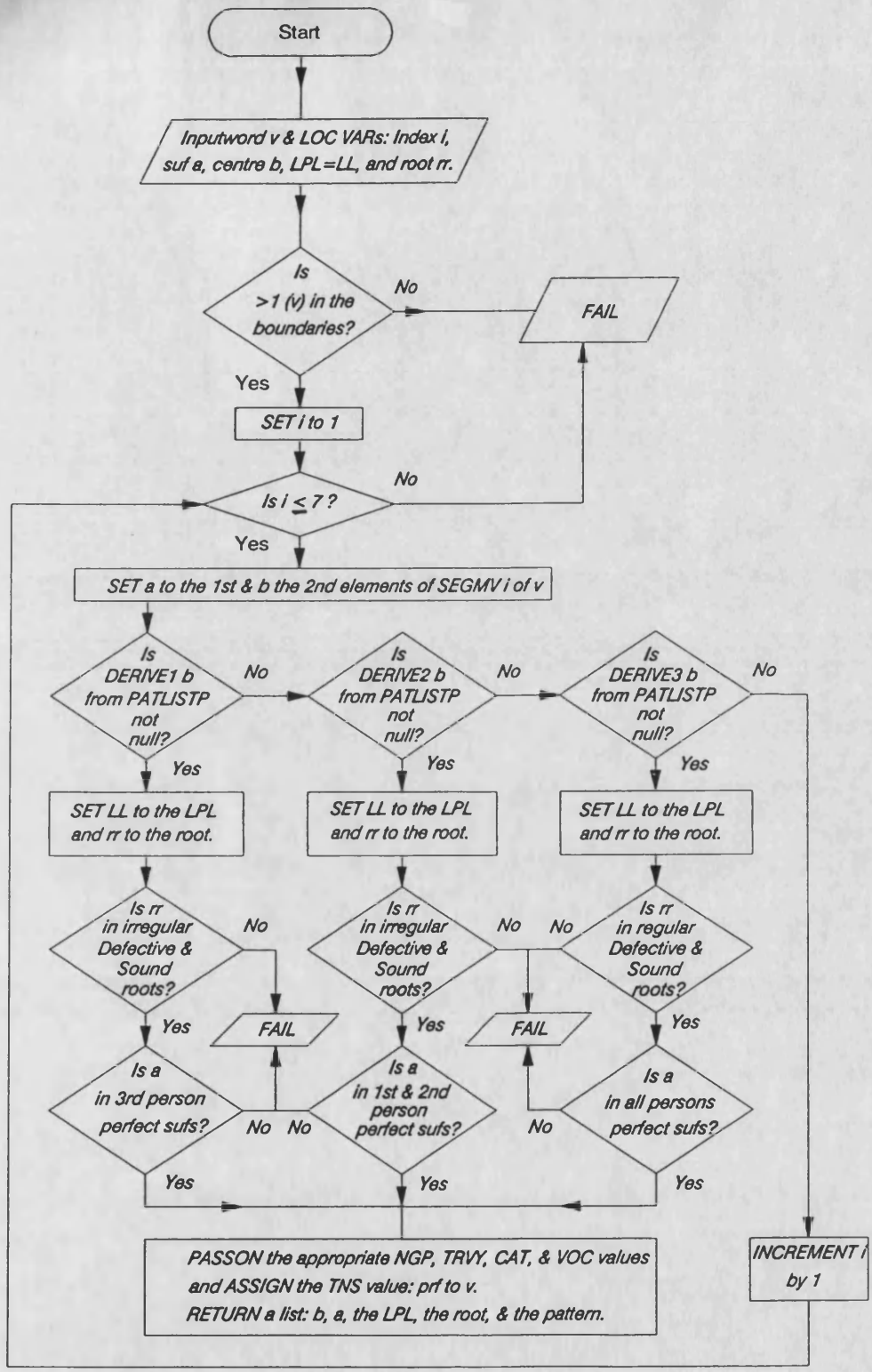


FIGURE 10: A Representation of the Perfect CF Recognition Procedure

DESCRIPTION TYPE	an abstract composite template structure.
STRUCTURE	*BASIC STRUCTURE:* a sequence of an obligatory prefix slot, an obligatory P/R combination slot and an obligatory suffix slot. *CONCATENATION ORDER:* given as P + P/R + S. *CF TYPES:* (a) *Sound*. (b) *Defective*. *EXAMPLES:* (a) P$acocuc$S. (b) P$acuwc$S.
CONSTRAINTS	*GRAPHOTACTIC CONDITIONS:* include boundary conditions such as Affix Selection Rules. *CF LENGTH:* unknown but ≥ 5. *P/R LENGTH:* the specified LENGTH. *P LENGTH:* L = [1]. *S LENGTH:* L = [1, 5]. *CF BOUNDARIES:* <1 (CF) = <1 (P), >1 (CF) = >1 (S). *P/R BOUNDARIES:* the defined boundaries. *P BOUNDARIES:* <1 (P) = {:/t/y/n}. *S BOUNDARIES:* >1 (S) = {a/i/o/u/y/!}. *LEXICAL SPECIFICATIONS:* (a) GCs (S) = GCs (P/R); and (b) ASRs (S) = ASRs (P/R).
ROLE	conjugate Verbs with all SPs in the imperfect TENSE using prefixes, P/Rs, and suffixes.
SIDE EFFECTS	possible NGP and MOOD disambiguation.

TABLE 27: A Formal Specification for an Imperfect CF

as ASSIGNed in the database, and the CATEGORY and TRANSITIVITY values as ASSIGNed by EXROOT; and it has to ASSIGN to *v* a TENSE value: *imperfect* and to RETURN a list of the prefix, the centre, the suffix, the LPL, the root, and the pattern.

Like MKPERFECT, MKIMPERFECT follows a context-sensitive method in DERIVation and property ASSIGNment using the GPL: "PATLISTI", which is a set of LPLs including roots ASSOCIATed with imperfect patterns. DERIVation is of three broad categories: DERIVE1 for irregular Defective and Sound roots with a third-PERSON form, DERIVE2 for irregular Defective and Sound roots with a first- and second-PERSON form and DERIVE3 for regular forms. A more detailed specification of PERSON values is given in Chapter 4, § II.1.1.2. For instance 'ay' is the valid 3 (M) S suffix allomorph for Defective roots such as {:b, :x, ns} in the passive VOICE, indicative and subjunctive MOODS and <a> is the valid 3 (M) S suffix allomorph

for those roots in the passive VOICE and jussive MOOD. Thus MKIMPERFECT needs to distinguish four types of context: root, affix, VOICE, and MOOD. The detailed specification of contextual property values is given in Chapter 4, § II.1.2.2. Following those specifications MKIMPERFECT ASSIGNS default or inherited values and new or contextual values, using the function PASSON. Where possible, MKIMPERFECT disambiguates homonymic property values using the concatenation context. The prefix and suffix boundaries of Table 27 restrict the application of MKIMPERFECT to inputwords that conform to the boundaries since as noted earlier constraining this application means significant savings on parsing time.

Unlike MKPERFECT, MKIMPERFECT is a procedure that looks left-to-right-to-left, i.e., centre-to-suffix-to-prefix but it is also a context-sensitive iteration for the prefix-LENGTH index [1] and the suffix-LENGTH interval [1, 5]. The procedure specifies consecutive indexes for that interval, progressing along the right end of the inputword each time the suffix index is incremented and the process of DERIVation for the centre and identification of suffix and prefix is attempted until a valid prefix-DERIVation-suffix is obtained, the *property heritage* ASSIGNed, and the result RETURNed. When the upperbound 5 is exceeded, the procedure is aborted. The exceptional cases, as in MKPERFECT, are tackled first then regular cases. Figure 11 below outlines the logical structure of MKIMPERFECT.

I.2.3.3 Graphotactic Filtering for Simple CFs

In Chapter 4, § II.3, we described Graphotactic Conditions which govern the boundaries of affixes when attached to other morphemes, such that there are valid and invalid sequences. Both MKPERFECT and MKIMPERFECT when analysing a Verb take account of the graphotactic nature of the affixes only in so far as conjugation rules are concerned but not ASRs. Rather, they are concerned with DERIVational constraints of regularity, structure of roots, and contextual properties. We can say that they overgenerate valid CFs in that they generate some CFs which, although they observe the conjugation rules, may violate the Graphotactic Conditions. This overgeneration was deliberately built into MKPERFECT and MKIMPERFECT in order to achieve a flexible design leading to more rapid processing. Adding the GCs would complicate the statement of the conjugational rules and slow down processing. We therefore propose the alternative strategy of *filters* which apply to the output of MKPERFECT and MKIMPERFECT, accepting it if it observes the GCs and rejecting it if it does not. Thus, we defined two *declarative* procedures: *FILTER1* and *FILTER2*. We understand by a *declarative* procedure one which is explicit, non-recursive, non-iterative, and which modifies data or previous output rather than executes operations.

FILTER1 operates on the output of MKPERFECT and *FILTER2* on the output of MKIMPERFECT. Both of these are context-sensitive in that they take into account the graphotactic

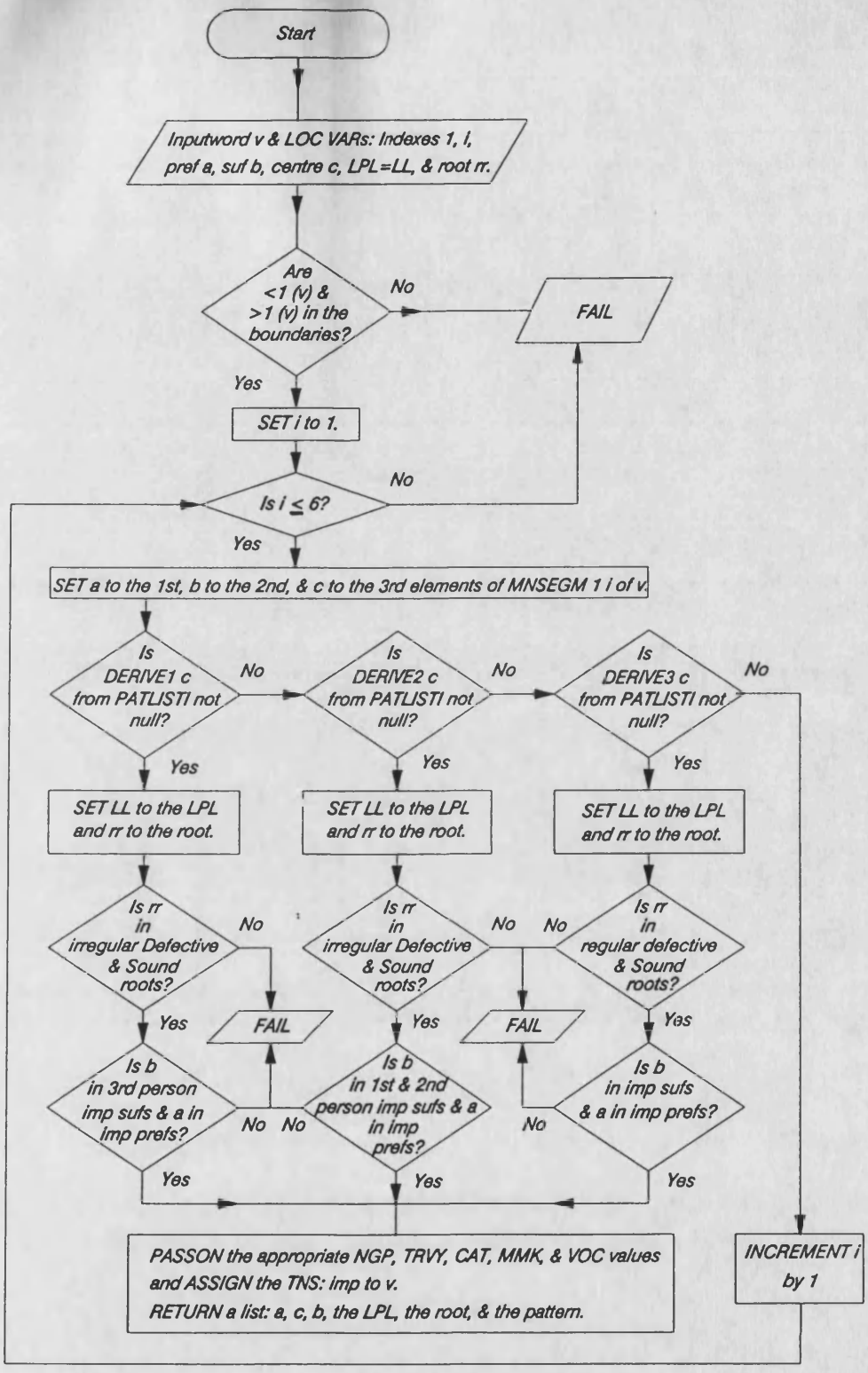


FIGURE 11: A Representation of the Imperfect CF Recognition Procedure

context at the boundaries between the centre and the suffixes. However, in addition to supervising the observation of GCs, FILTER2 has another function. This is the resolution of certain types of Imperfect CF homonymy. In particular, in Chapter 4, § II.2.4, we noted that certain Defective CFs such as: 'tacociyna' and 'tacocuwna' could have two valid SEGMENTations: 't$acociy$na'/'t$acoc$iyna' and 't$acocuw$na'/'t$acoc$uwna' respectively, since all of the affixes: {na, uwna, iyna} and all patterns: {acociy, acocuw, acoc} are valid. However, it is the second of each of the two SEGMENTations that is desired since they distinguish the suffixes: {iyna, uwna} which will enable the appropriate NGP and MMK ASSIGNment. Nevertheless, MKIMPERFECT RETURNS the first of each of these SEGMENTations, since it operates on smaller segments first, incrementing the LENGTH index progressively: thus 'na' (smaller than 'iyna' and 'uwna'), is found first and is RETURNed as the result as soon as it conforms to the conditions specified in MKIMPERFECT. The task of FILTER2 is to catch that result and perform a slight modification or *reanalysis* of the SEGMENTations, generating the suffixes 'iyna' or 'uwna', a new centre MATCHing the new pattern 'acoc' not 'acociy' or 'acocuw' and ASSIGNing the NGP and MMK values of the new suffix, thereby resolving this kind of homonymy as early as possible.

The function *VPARSE* monitors the output FILTERS 1 and 2 so as to RETURN either a result, which is the output of MKPERFECT or MKIMPERFECT, or an error message or a FAILure in this way:

- IF FILTER1 FAILS, then RETURN the result of FILTER2,
- ELSE RETURN the result of FILTER1.

This ensures that VPARSE does not RETURN an error message from FILTER1 if FILTER2 has a successful result. This is because an error message which is a string is treated in CL as a non-NIL value. VPARSE can then be called to perform V-Processing without specifying FILTER1 or FILTER2.

I.2.3.4 Categorial and TRANSITIVITY MODIFICations

In Chapter 4, § II.2.6, we outlined categorial homonymy cases for Simple CFs. We also specified that the TRANSITIVITY value of a Verb depends on its VOICE. However, the VPARSE procedure does not take account of either of these facts. It would add unnecessary complication to incorporate such special cases in that procedure. Instead, a better solution is to define a new procedure that we will call *TVACT1* to *act* on the TRANSITIVITY values of a given CF and tackle categorial homonymy.

The TVACT1 procedure will have the task of monitoring the output of VPARSE and inspecting the VOICE of each CF as it is coming out of VPARSE. If the VOICE is active

then TVACT1 specifies a number of contexts detailed in Chapter 4, § II.2.6, where categorial homonymy arises and raises flags for these by ASSIGNing homonymic categorial values. These homonymic cases also necessitate the ASSIGNment of Nominal property values specified in Table 20 but not ASSIGNed in VPARSE. TVACT1 also makes use of this opportunity to disambiguate the TRANSITIVITY values ASSIGNed by VPARSE. If the VOICE is passive then the TRANSITIVITY value is DEMOTed to a lower value. The definition of TVACT1 is a declarative procedural algorithm which takes a Simple CF as an argument and does not perform any SEGMENTation or processing on it, rather it RETURNS the same output structure as VPARSE but MODIFIes the property-list structure generated by VPARSE.

I.2.3.5 Accusative-Pronoun Stripping and Further Categorial Disambiguation

As defined so far, TVACT1 can recognize only Simple CFs. We can deduce from the description of Chapter 4 a formal specification for Complex CFs which we give in Table 28 below and which will allow us to develop a procedure that is able to recognize Complex CFs.

We start with Simple CFs with clitic Accusative Pronouns, i.e., Referential CFs. We define a procedure called *XPARSE*, which using the above specifications, takes an inputword x as an argument and checks first to see if TVACT1 applies to the whole of the inputword; if otherwise, it proceeds right-to-left, stripping and inspecting consecutive chunks off the end of the inputword until it is able to locate them in the enclitic Pronoun dictionary. When it does find a Pronoun, XPARSE attempts to obtain a TVACT1 result for the REST of the inputword, and if it does, it MODIFIes CATEGORIAL values for homonymic CFs as specified in Table 24; otherwise, it performs simple boundary transformations at the CF-Pronoun junctures. This is to enforce Graphotactic Conditions of affixation as described in Chapter 4, § II.4.3. These transformations are simple operations such as supplying the character <|> which should be DELETed in the inputword upon affixation of CPs in order to observe the affixation conditions. The insertion of <|> has the effect of allowing a MATCH between the inputword and the database form and therefore causing TVACT1 to RETURN a non-NIL result. However, if the inputword itself has that character at the boundary then it is rejected as illegal.

XPARSE also ASSIGNS REFERENCE values to x if x contains a Pronoun. It is important to distinguish such Referential CFs in anticipation of syntactic rules (which we discuss further in Chapter 12). Here the *property heritage*, i.e., the set of property values inherited from Simple CFs, is also passed on from the Simple to the Complex CFs. In this case, the property heritage is called *VPROPLIST* which is the set of property lists for NGP, CAT, TNS, VOC, TRVY, MMK, and REF. The transfer of this heritage is performed by *PASSON2*, which is the same as PASSON, except in one respect. Whereas PASSON transfers default values directly from

DESCRIPTION TYPE	an abstract composite template structure.
STRUCTURE	*BASIC STRUCTURE:* a sequence of optional affix slots: F and CP, and an obligatory CF slot. *CONCATENATION ORDER:* given as (F) + CF + (CP). *CF TYPES:* (a) *Perfect CF*. (b) *Perfect CF$CP*. (c) *Imperfect CF*. (d) *F$Imperfect CF*. (e) *F$Imperfect CF$CP*. (f) *Imperfect CF$CP*.
CONSTRAINTS	*GRAPHOTACTIC CONDITIONS:* include boundary conditions such as Vocalic Compatibility. There are also TENSE and MDG conditions. *CCF LENGTH:* unknown but $\geq$ 4. *CF LENGTH:* the specified LENGTH. *F LENGTH:* L = [2]. *CP LENGTH:* L = [2, 5]. *CCF BOUNDARIES:* *if* F then <1 (CCF) = <1 (F), else <1 (CCF) = <1 (CF); and *if* CP then >1 (CCF) = >1 (CP), else >1 (CCF) = >1 (CF). *CF BOUNDARIES:* the defined boundaries. *F BOUNDARIES:* <1 (F) = {s/l}. *CP BOUNDARIES:* >1 (CP) = {a/i/o/u/y/l}. *LEXICAL SPECIFICATIONS:* (a) *if* F then, TENSE, MOOD (F) = TENSE, MOOD (CF); and (b) *if* CP then, VCo (CP) = VCo (CF).
ROLE	provide a template for the realization of CFs with or without attached F, with or without attached CP.
SIDE EFFECTS	possible NGP, MOOD, and categorial disambiguation.

TABLE 28: A Formal Specification for a Complex CF

the database without removing them, *PASSON2* performs a certain amount of *garbage collection* by removing the contextually ASSIGNed properties of smaller units as it passes them on to larger ones. In this way, we avoid cluttering up the property list by leaving properties that are no longer needed lying around.

XPARSE is an iterative procedure which performs an iteration for the Pronoun-LENGTH

interval [2, 5] specified in Table 28 and keeps incrementing the index until the upperbound is reached and all the SEGMENTation and search possibilities are exhausted. The procedure RETURNS the inputword x for non-Referential CFs and a list of the Simple CF and the Pronoun for Referential CFs. To cut down on the amount of searching done, XPARSE also specifies constraints on the nature of final characters of x using the boundaries specified in Table 28 and input that does not conform to this entry condition is filtered out before parsing is attempted. Figure 12 below illustrates the XPARSE iterative right-to-left method for Complex CF recognition.

I.2.3.6 Graphotactic and TRANSITIVITY Filtering for Complex CFs

XPARSE does not take account of the TRANSITIVITY and Vocalic Compatibility conditions (VCo) in Chapter 4, § II.4.2 and § II.4.3. This is because XPARSE was concerned mainly with the identification of the component morphemes of CCFs, and the ASSIGNment of the appropriate property values once these morphemes were identified. In order to keep the statement of the routines achieving these tasks of recognition and ASSIGNment to a minimum of simplicity, it is better to account for the above VCo conditions separately, rather than include them in the above recognition routines, which would complicate their statement. Therefore, we defined *FILTER3* to monitor the output of XPARSE and enforce the observation of these conditions.

The task of FILTER3 is then to inspect all Referential CFs. In anticipation of syntactic analysis, these are divided into two groups:

1) *collocutive* CFs, which are CFs with clitic first- and second-PERSON Pronouns, and

2) *exlocutive* CFs, which are CFs with clitic third-PERSON Pronouns.

We can anticipate that collocutive CFs will have optional antecedents while exlocutive CFs will require antecedents. However, before FILTER3 performs that distinction, it examines the TRANSITIVITY value of the CF in order to exclude intransitive and ptransitive1 CFs from being attached with CPs. Collocutive Referential CFs are then allowed through as their Pronouns do not have allomorphic variation but exlocutive CFs are forced to have their pronominal variants comply with the compatibility conditions of affixation. Like FILTERS 1 and 2 and TVACT1, FILTER3 is a declarative context-sensitive procedure which does not carry out processing but monitors output to enforce conditions that are too complicated to incorporate in the procedures that it invokes.

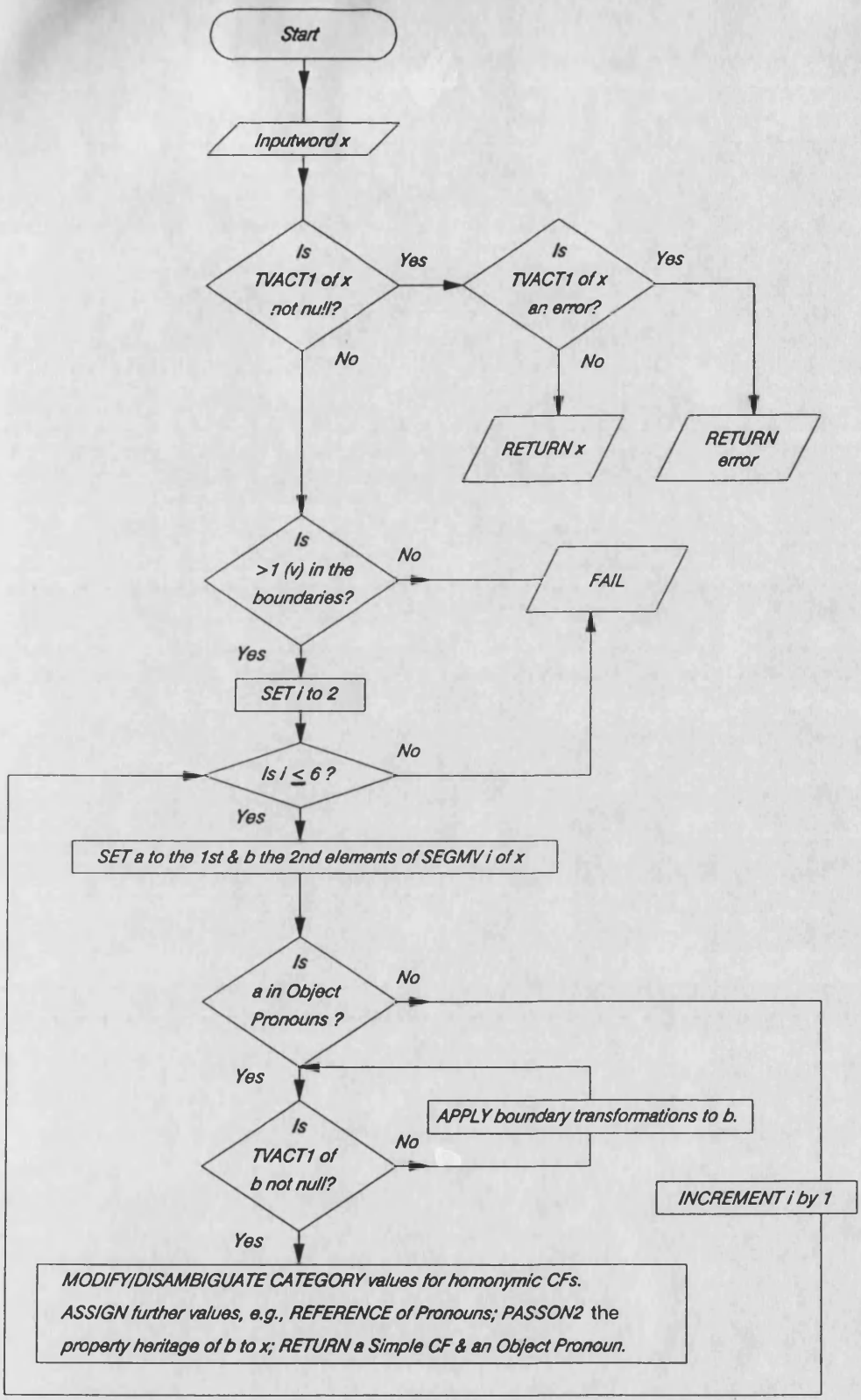


FIGURE 12: A Representation of the Complex CF Recognition Procedure

I.2.3.7 Futurization and MOOD Disambiguation

As specified in Table 28 a Complex CF may carry, besides an Accusative Pronominal suffix, a Future Particle prefix. We will call such a CF a Future CF. In order to recognize a Future CF we defined the LR procedure: *FUTURIZE*. This procedure takes an argument x for Complex CF and checks first to see if FILTER3 applies to the whole inputword x, otherwise, it strips a prefix chunk of LENGTH 2 off the word as in the specification. If this prefix MATCHes one of the two Future Particles, then if FILTER3 of the REST of the word is successful, then the CF generated by FILTER3 must conform to the lexical specification or MOOD GOVERNMENT of Future Particles. Remember that the Particle 'sa', "going to", requires a CF that has imperfect TENSE and indicative MOOD and the Particle 'li', "in order to, ...", requires a CF that has imperfect TENSE and subjunctive MOOD. CFs that have homonymic values including the indicative or subjunctive and that are said to be homonymic with the indicative or subjunctive are also accepted and their value can now be disambiguated since, given the Future Particle prefix, only one of the homonymic interpretations is then possible. In some cases specified in Chapter 4, § II.4.4, the disambiguation of MOOD values also allows the disambiguation of NUMBER values.

After the constraint verification and disambiguation, the property heritage of the Complex CF is transferred to x (i.e., the Future CF), and a new CATEGORY is ASSIGNed to x. This CATEGORY is a composite of P for Particle and the CATEGORY of the Complex CF which is either VB, VR or some other ambiguous CATEGORY as before, giving PVB, PVR, etc. FUTURIZE is a context-sensitive left-to-right procedural method for processing a Future CCF which does not involve any iteration since Future Particles have a constant LENGTH. However, the boundaries of these prefix Particles: {s, l} are used to filter out input that does not conform to them. The value RETURNed by FUTURIZE is the inputword x for non-Referential CFs and a list of the CF and the Accusative Pronoun for the Referential CF. This procedure is illustrated in Figure 13 below.

II SYNTHESIS OF THE V-COMPONENT

II.1 THE LOGICAL STRUCTURE OF THE V-COMPONENT

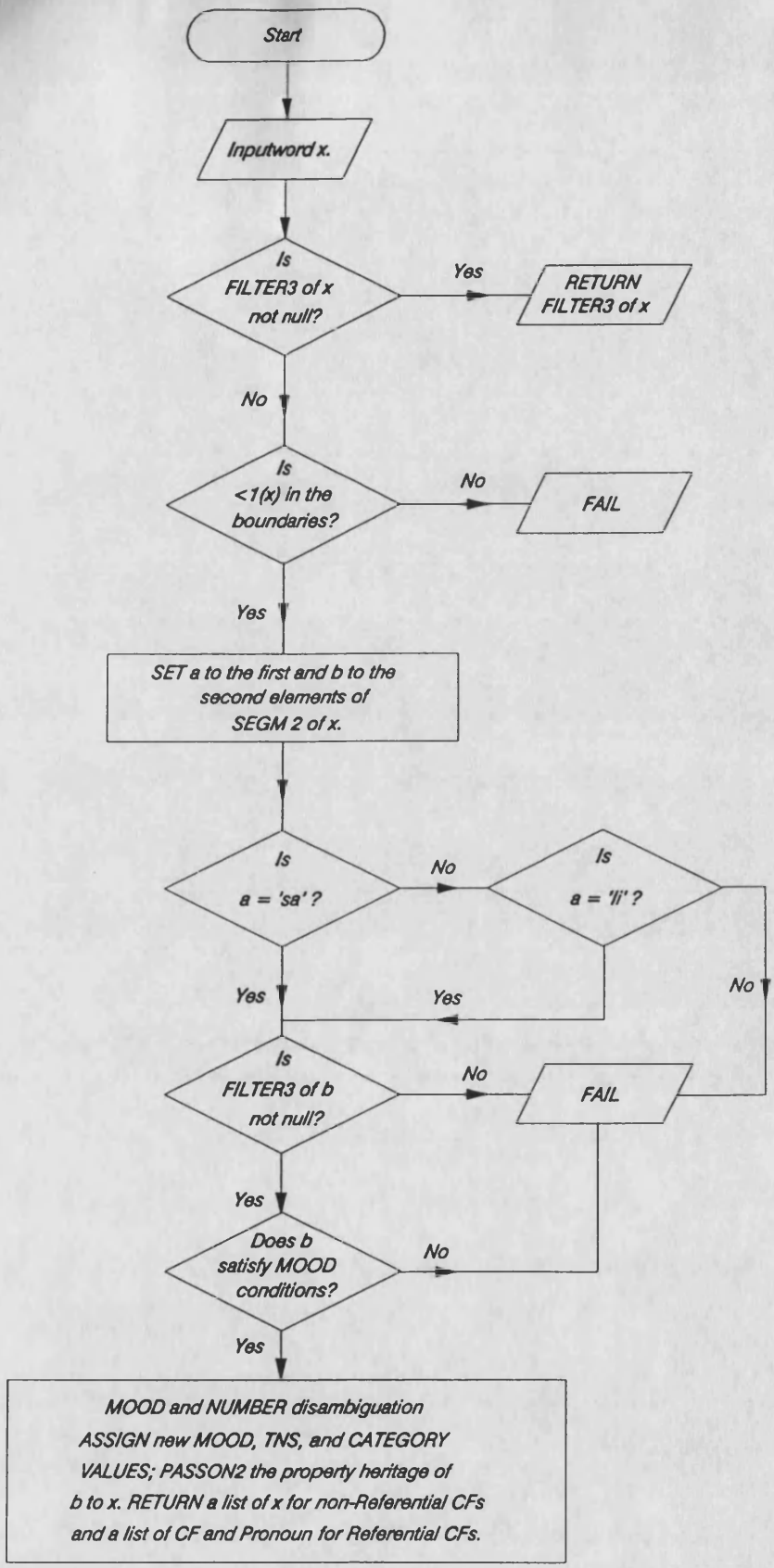


FIGURE 13: A Representation of the Future CCF Recognition Procedure

II.1.1 The Structure of the V-Complex

In Appendix A, we gave a list of Conjugation Forms, or CFs, divided according to a V-Typology outlined in Chapter 4, § I. In Chapter 4, we used a collapsing method to generate pattern representations for sets of Basic and Augmented, Sound and Defective roots with regular and irregular conjugations but these generalizations were then refined to take account of homonymic phenomena.

The linguistic analysis of V-Complexes defined their morphological constitution and distinguished obligatory *inner* affixes inside Simple CFs and optional *outer* affixes inside Complex CFs, all these affixes having differing degrees of allomorphic variation for such affixation. The analysis detailed Graphotactic Conditions of compatibility, syntactic conditions concerning TRANSITIVITY and lexical specification or MOOD GOVERNMENT, DERIVation conditions concerning regularity of conjugation and root types, and Affix Selection Rules relating to root and suffix types. Each morphological unit was observed to carry a bundle of feature values, called its *property heritage*, that affect the set of features of the larger unit where it is embedded. The values including ones for NUMBER, GENDER, PERSON, TRANSITIVITY, VOICE, CATEGORY, TENSE, and MOOD were divided into default and contextual values and their ASSIGNment was done initially on the default basis for unique values, whilst contextually homonymic values were either disambiguated or given flags. In other cases neutral values were used.

We saw that Cambridge LISP provides us with facilities for the direct representation of the described data structures in a structured database directly accessible by subsequent programs. By specifying sets of dictionaries, we defined constraints on database access anticipating dictionary-dependent rule application at run time. This is in fact the only type of condition on database access, and is designed to guide the application of rules in order to reject DERIVations and structures that do not comply with these rules. In particular, the distinction of regular and irregular, and of causative and intensive dictionaries can be used to guide DERIVations.

The analysis we made of the CFs then enabled us to define in this Chapter formal specifications for them, which are concerned with types of CF and their constituent structure, the affixation of Future Particles and Accusative Pronouns to such CFs, as well as with the TRANSITIVITY, GOVERNMENT, and Graphotactic Conditions under which such affixation is allowed. These specifications also detailed constraints on the realization of morphemes in terms of their boundaries and their LENGTH. This structure of the V-Complex is represented in Figure 14 below:

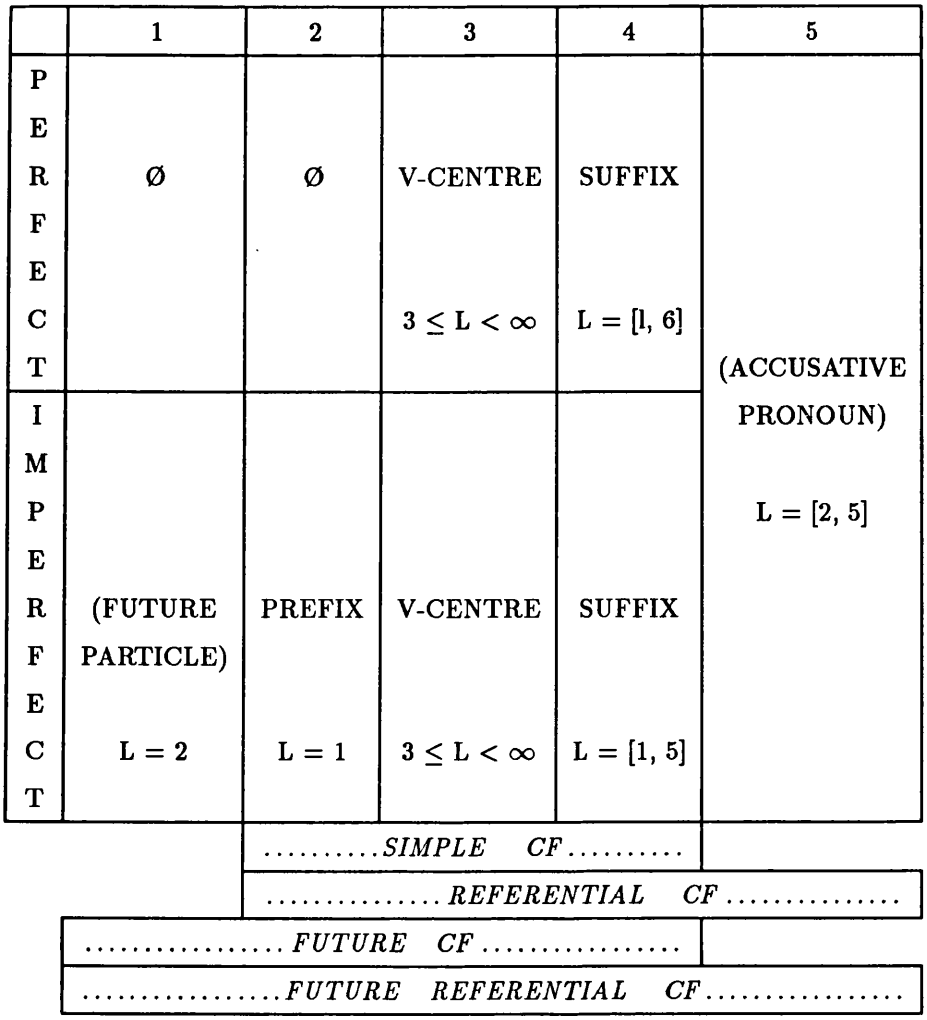


FIGURE 14: A General Model for the V-Complex in M.S.A.

II.1.2 The Structure of the V-Processor

The computational procedures of the V-Processor described in this Chapter implement the linguistic description given in Chapter 4. The boundary specifications are implemented as entry conditions on the operation of higher procedures in the processor. The LENGTH specifications are implemented as numerical intervals with a threshold at which lower routines are invoked and an upperbound at which a procedure is exited. The affixation conditions are implemented as procedural filters which monitor the output of lower routines and reject it upon infringement of any of the conditions. The lexical specification or MOOD GOVERNMENT rules are implemented directly as requirements on the segment in question. Boundary variations are handled by simple yet effective use of transformations which supply characters that are obligatorily DELETed in the input, thereby simulating an exact MATCH with the database. In the course of a given program run, each context is noted and the appropriate feature values ASSIGNed; and, as larger concatenations are discovered, the property heritage thus ASSIGNed is MODI-

FIed if need be and then transferred from the smaller to the larger units. The identification of such contexts can be seen as falling within this logical framework:

<SEGMENTation>,
<DERIVation>,
    <**case**: root context>,
        <**subcase**: VOICE context>,
        <**subcase**: suffix context>,
        <**subcase**: MOOD context>,
    <**action**: property ASSIGNment>,
    <**result**: output structure>,
<**repeat** SEGMENTation>.

II.1.3 The Architecture of the V-Processor

The V-Processor itself is a structured program, consisting of several interrelated routines. The most basic of these are small and fast recursive subroutines that carry out the most frequent operations of the program such as SEGMENTation, MATCHing, and root EXTRACTion. Above these routines are placed intermediate recursive subroutines that coordinate DERIVational operations by invoking root EXTRACTion and MATCHing after the retrieval of pattern lists. The main procedures are context-sensitive operations that fall into two categories: primary iterative or non-iterative procedures whose function it is to perform recognition, property ASSIGNment and ambiguity resolution; and secondary procedural operations or filters which declare TRANSITIVITY and Graphotactic Conditions to be observed and whose function it is to enforce these conditions. These filters do not access the database, whereas the primary procedures and the subroutines do. Each function in the V-Processor has a specified number of arguments, besides any given number of LOCAL VARIABLES used where necessary, one or more objectives ASSOCIATed with it, and a value or output result RETURNed.

The analysis in this Chapter describes the various techniques used, such as iteration or recursion; and these depend on the task required. Thus, each lower part or MODULE of the V-Processor is designed to carry out partial tasks, while higher modules coordinate the final output. This modular system architecture makes for a flexible and efficient system that is easy to update and where it is easy to pinpoint mistakes if they occur. The logical layout or architecture of the system is shown in Figure 15, which illustrates the organization of the V-Processing modules, the operations they perform and points at which they access the database.

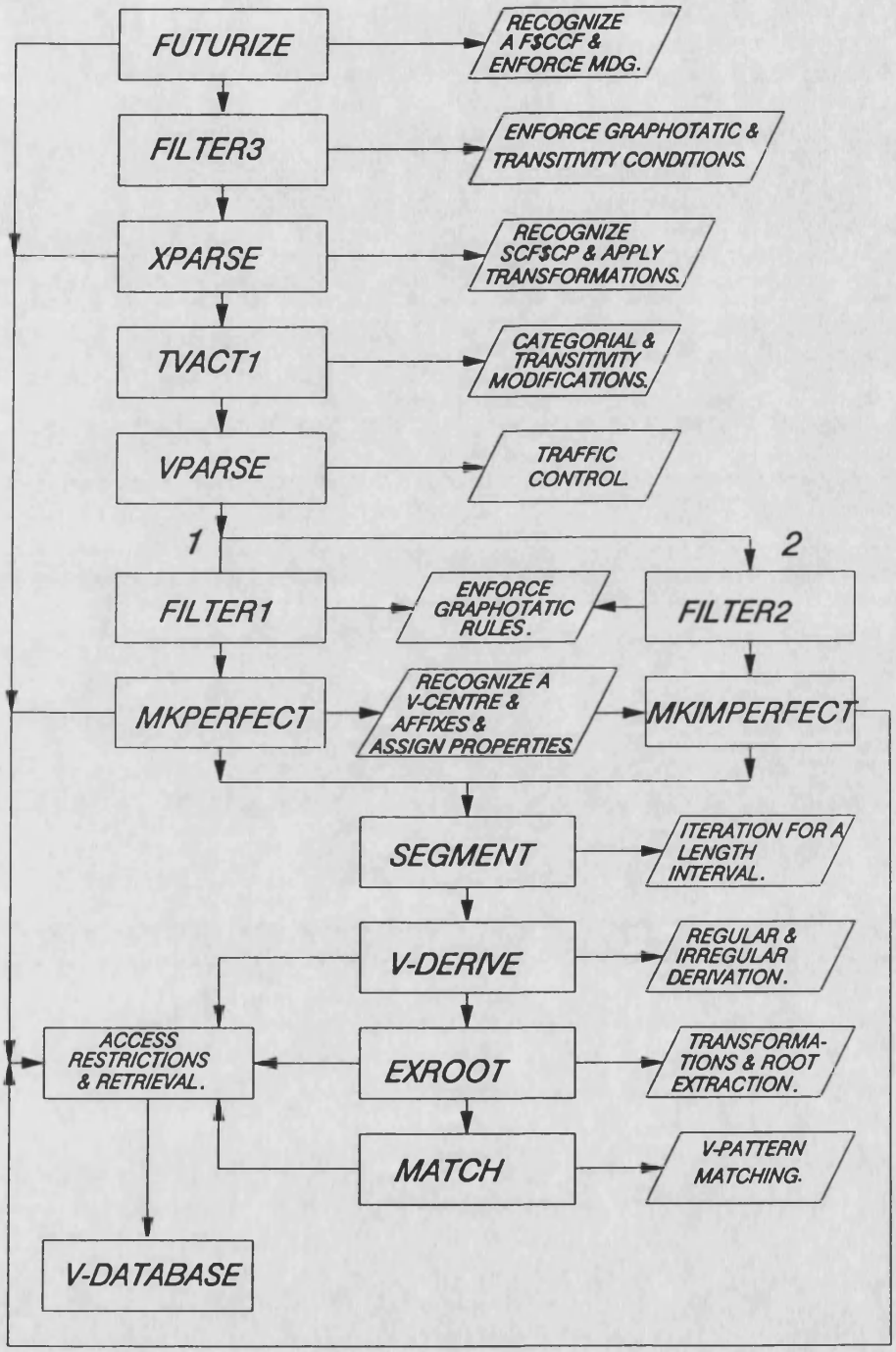


FIGURE 15: The Architecture of the V-Processor in TUNIS1

II.2 THE SEARCH PHILOSOPHY OF THE V-PROCESSOR

II.2.1 Converting M-Trees to V-Goal Trees

In Chapter 4, § II.5, we were able to define a set of MS rules that can be represented by the abstract objects called M-Trees, or M-Markers. In order to use the M-Trees in the task of V-Recognition and in particular in order to devise an optimal strategy for such recognition, the most efficient computational method is to convert such M-Trees to Goal Trees which allow a natural progression towards search strategies. In Chapter 2, § III.1, we have provided rules for the conversion of M-Trees into G-Trees.

The V-Goal Tree provides another natural means to account for the linguistic data, by allowing the direct representation of morphological precedence rules as nodes, and of morphological conditions as constraints on the satisfaction of the nodes. Figure 16 shows a G-Tree for the V-Complex in M.S.A. and illustrates the ordered search paths followed by the V-Processor.

II.2.2 The Search Strategy and the Flow of V-Parsing

Figure 16 shows a Goal Tree with subtrees to be solved. The whole tree is solved if one of its *or* subtrees is solved. An *or* subtree is solved if one of its *or* nodes and all its *and* nodes are solved. If there are constraints on a node, then it can be solved only if the constraints are satisfied. However, what is the optimum path that we can follow in solving the tree? Given the arguments outlined previously (Ch. 2, § III.2.3.1) and relating to the linguistic bases to the search strategy and to the direction of parsing, we want our parser to start with the simplest and most likely possibilities, first, and to progress towards more complex ones, proceeding in a right-to-left direction whenever possible.

In the case of the V-Processor, the simplest possibility is a Perfect CF with only two segments: a V-Centre and a suffix. The processor attempts to apply MKPERFECT first and inside MKPERFECT performs an iteration for the LENGTH of the suffix starting in a left-to-right manner by attempting a DERIVation for the centre and in a right-to-left manner by considering the smallest suffix first. If MKPERFECT FAILS then MKIMPERFECT is called. Inside MKIMPERFECT the simplest route is left-to-right-to-left. A prefix of constant LENGTH 1 is stripped and a DERIVation is attempted for the centre after stripping suffixes of LENGTH 1 to 5 starting with the smallest suffix. If MKIMPERFECT FAILS then the next larger concatenation: a Complex CF is attempted. We have seen that a Complex CF is a Referential CF, a Future CF or a Future Referential CF. Using the right-to-left approach,

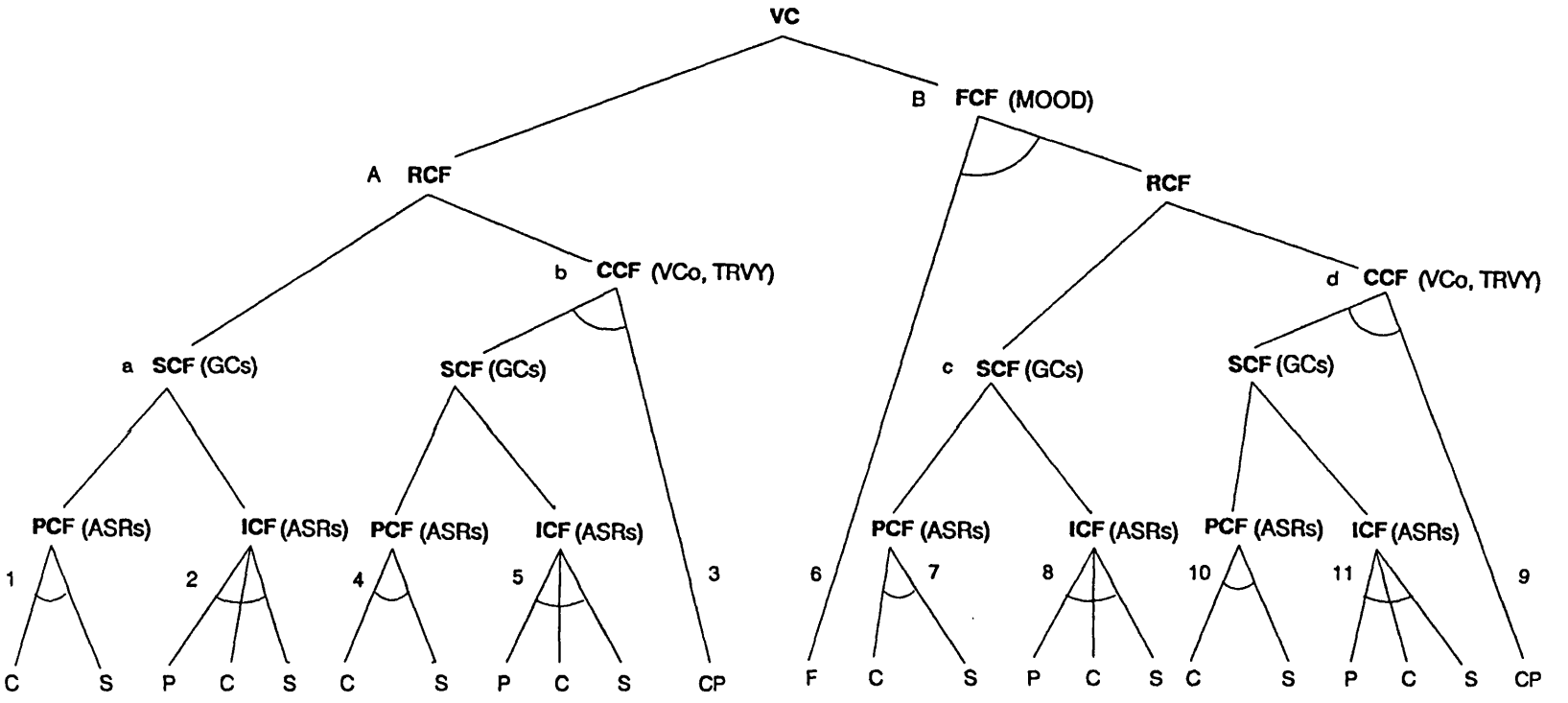


FIGURE 16: Search Paths and Constraints in a G-Tree for the V-Complex in M.S.A.

the processor invokes XPARSE and proceeds to strip off Accusative Pronominal suffixes of LENGTH 1 to 5 until one is found or the search FAILS. If one is found, then the processor attempts MKPERFECT and then MKIMPERFECT again. If XPARSE FAILS, then the processor invokes FUTURIZE, which strips off a Future Particle prefix of constant LENGTH 2 and attempts MKPERFECT, MKIMPERFECT, and then XPARSE, which after searching for a pronominal suffix will attempt MKPERFECT and then MKIMPERFECT; if they FAIL the V-Processor FAILS. At each exit of these procedures, TRANSITIVITY, MOOD, and Graphotactic Conditions apply as constraints on satisfying a given node. These paths and mechanisms are illustrated in Figure 17 below.

The exhaustive method followed by the V-Processor also means that it may visit repeated states more than once. However, this method is more efficient in this case than other approaches. For instance, a purely depth-first method requires expensive saving of all the paths in order to go back to decision points to restart just in case a wrong choice is made. Since a partial tree, or subtree, may solve the whole tree, the breadth-first approach is not an adequate one either.

The result of each V-Parse is a flattened tree and not all the nodes are RETURNed in output structure. Thus, Figure 14 shows possible hierarchical bracketed structures of the form: [(F) [(P) Ce S (CP)]]. However, the V-Processor actually generates only single or unified categorial symbols, such as: VB for Simple Perfect and Imperfect CFs, VR for Referential CFs, PVB for a Future CF, and PVR for a Future Referential CF: FPCeSCP.

We have already given an illustration of a complete G-Tree in Figure 16. We have also given some sample morphological V-Parses by TUNIS1 for further illustration of the procedures of the V-Processor. In these samples, (listed in Vol. II, App. D, § A), the global function *MKTRUNK*—which calls FUTURIZE as a subprogram—is embedded in the function call TIME, and three results are given together:

- the CPU time taken in parsing a given inputword simply noted as input;
- the output structure called the value (of the parse); and
- the complete property list for the output structure where inapplicable property values are noted as a Ø.

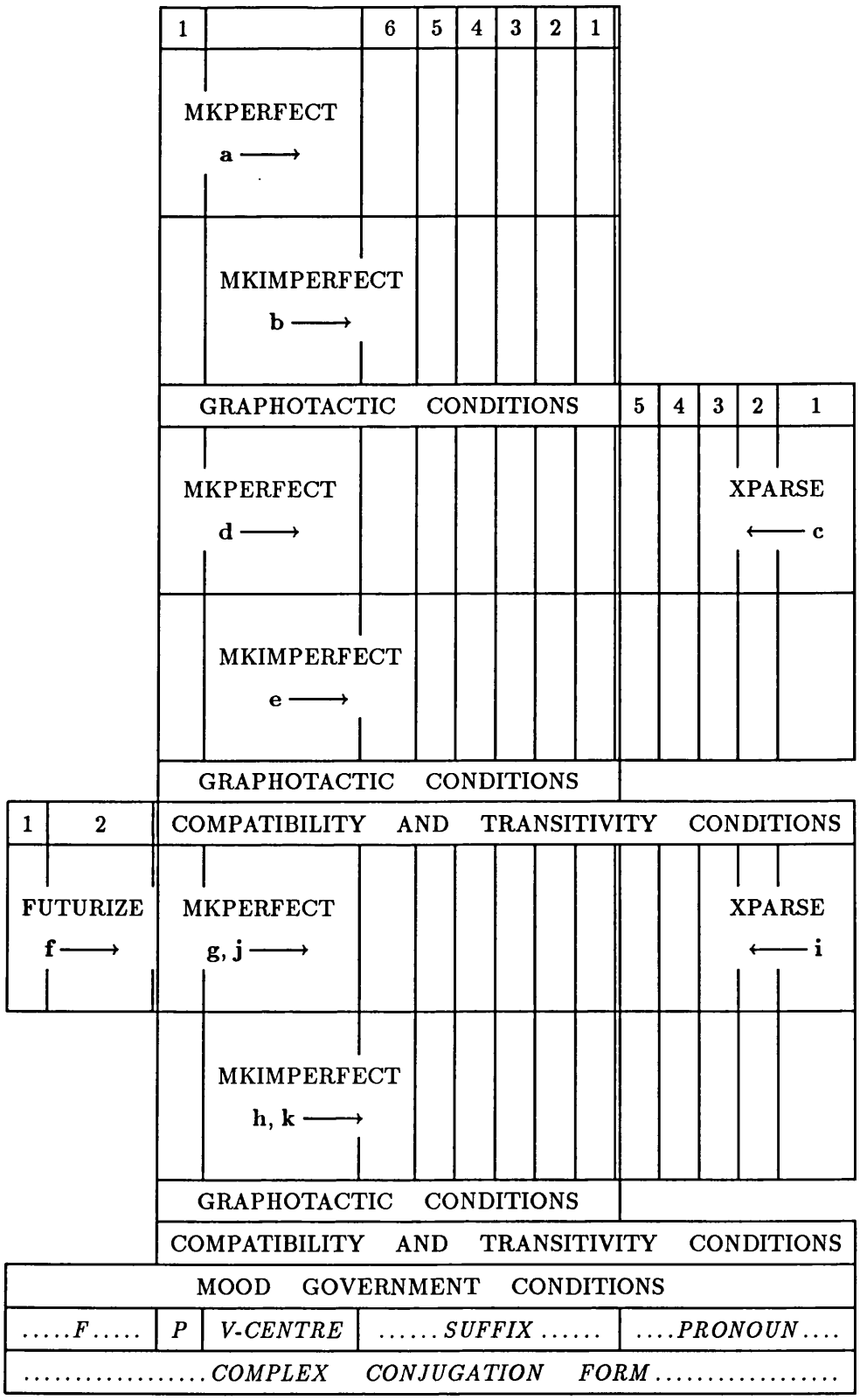


FIGURE 17: A Chart for the Direction of Parsing in the V-Processor

==0==<*****>==0==

Part III

THE NOMINAL COMPLEX IN M.S.A.

Chapter 6

A FORMAL MODEL OF THE N-COMPLEX

INTRODUCTION

In this Chapter we start with a definition and a linguistic précis of the structure of the *Nominal* in M.S.A. We describe, in § I, a typology of the Nominal using the following structural criteria: the structural parallels between Verbs and Nominals—especially with respect to *Verbals* (defined below)—and the graphological and morphological pattern and radical structure of the Nominal. This allows us to define the Arabic Nominal system.

We then delineate the scope of the analysis of Nominals undertaken in this study and set up a list of derivational patterns for each CATEGORY of Nominal (in App. A, § B). These patterns are given in three different derivational groups: *Masculine-Feminine* (*MF*), *Masculine Only* (*MO*), and *Feminine Only* (*FO*); and in two different types: a *Base* type and a *Broken Plural* (*BP*) type. The groups and types of pattern allow us to define more formally (in § II) the precise constituent structure of the Nominal in M.S.A. By studying the structure of Nominal derivations we are able to make generalizations about the processes of generation of NUMBER and GENDER categories. These processes include the generation of the masculine singular, *feminization, dualization,* as well as (masculine and feminine) Regular and Broken *Pluralization*. We also make generalizations about the inflection of the Nominal for CASE. Inside the Nominal structures generated in this way—which we will call *Simple Nominal Forms*, or *NFs*—we identify Nominal stems and patterns as well as affixes, which have different distribution and allomorphic variations for *Sound* and *Defective* Nominals (as defined below). For all

such NFs, we detail features and feature values for NUMBER, GENDER, PERSON, VOICE, TRANSITIVITY, CASE, FLEXION, HUMANITY, PLURAL CLASS (PLC), (graphological) TYPE, and CATEGORY. The definition of Simple NFs allows us to expand their structure to cover Complex NFs that are *definitized* via Determiners and Genitive Pronouns. The complex structures themselves carrying DEFINITENESS- and REFERENCE-feature values can then be embedded in Prepositional NFs.

However, it turns out that our generalizations have to be constrained, in § II.2, by the application of various affixation conditions that govern the juxtaposition of Nominal structures. Further, the generalized rules have to be refined in order to take account of homonymic phenomena in the distribution of property values, and in structural SEGMENTation. Fortunately, the specification of affixation conditions and contexts allows the disambiguation of certain cases of homonymy. This specification allows us, in § II.3, to deduce a body of formal structural rules—called *Annotated MS rules*—in order to generate "all and only" valid Nominal M-Trees. The linguistic description and formal account of Nominal structure in this Chapter, together with the formal structural rules and affixation conditions, enable us to construct a formal morphological model of the N-Complex and its logical structure, a model that the N-Processor can harness in order to parse such Complexes optimally and efficiently.

I A LINGUISTIC DESCRIPTION OF THE NOMINAL IN M.S.A.

I.1 A TYPOLOGY OF THE NOMINAL: DERIVATIONAL PROCESSES IN M.S.A.

Like other Arabic terms for parts of speech, the *':isom'*—usually translated as "Noun"—has traditionally been given only loose definitions. For example, SHARIYF, 1979: 17, gives the following semantic definition: "a Noun is that which has a meaning of which TENSE is not an integral part, or that which can be an argument or a predicate". A more functional definition is given by ALMAXZUWMIY, 1966: 27–28, "Nouns represent meanings that cannot otherwise be expressed, and have features that distinguish them from other categories, such as 1) definiteness 2) gender, and 3) number".

Definitions of the English word *noun* are similarly vague. For instance, CRYSTAL, 1980: 244, claims that: "nouns are ITEMS which display certain types of INFLECTION (e.g. of CASE or NUMBER), have a specific distribution (e.g. they may follow PREPOSITIONS but not, say, MODALS), and perform a specific syntactic function (e.g. as SUBJECT or OBJECT of a sentence)". WINOGRAD, 1983: 52, summarizes the traditional definition of the Noun as that part of speech which represents a "person, place, thing, or concept".

However, such definitions remain loose in mixing various syntactic, functional, and semantic criteria. SHARIYF's definition, in an attempt to include so-called *Participles* (a CATEGORY to which we return below) manages to include Verbs—as predicates—in the definition of *Noun*. Similarly, ALMAXZUWMIY fails to notice that NUMBER and GENDER can also be features of the *Verb*; while CRYSTAL's definition, if used to define the Arabic '*:isom*', will fail to cover *Participles*. Thus, from the outset, we can discern a potential source of confusion and controversy, namely, that in M.S.A., the term '*:isom*', or "Noun", is used to cover two broad types, the *Primitive Substantive*, or '*:isom ja/mid*', and the *Derivative*, or '*:isom musvotaqq*'.

a. *Primitive Substantives* are those lexical items whose derivation is morphologically and semantically unpredictable, i.e., they cannot plausibly be related to another CATEGORY in (stem) meaning or in pattern structure. Further, Primitive Substantives are items whose stem cannot be used to derive a Verb which has the same core meaning as that conveyed by the Substantive itself, and which are used 1) for humans and 2) for non-humans.

b. *Derivatives* are those lexical items whose stems and patterns are related in meaning and pattern structure to the CATEGORY of Verbs, and which include *Infinitives, Qualificatives*, and *Derived Substantives*:

i. We use the term *Infinitive* here as a translation for the Arabic term *'mas;odar'*, literally, "source", to refer to those lexical items which are derived using a V-pattern and preserving the meaning of the Verb root, but changing the V-Pattern according to predetermined simple transformations; and which, unlike Verbs, have no pragmatic-time reference.

ii. *Qualificatives* are defined as those lexical items used to characterize an entity by a particular quality, or attribute. They also have a pragmatic-time reference, which is generally the present time, but can be the future or the past, if the Qualificative is modified by another CATEGORY, such as an Auxiliary, e.g., 'ka|na', "was".

iii. *Derived Substantives*, are lexical items used to describe variables in a given situation, such as the *location*, *time*, or *instrument* of an action whose meaning is denoted by a Verb that has the same root as the Derived Substantive.

The derivational processes referred to immediately above may follow idiosyncratic patterning, but are generally subject to regular patterning, and/or simple transformational rules. Table 29 below shows a detailed classification of the Nominal using the derivational criterion to illustrate its morphological pattern structure in M.S.A.

NOMINAL TYPE		DERIVATION & PATTERN TYPE	DERIVATIONAL PATTERN & PROCESS	EXAMPLE
Primitive Substantive		unpredictable derivation; (includes PNs & Common Nouns)	several patterns, e.g., cawa|c.	'jawa|d', "horse" 'zayod', "Zayd"
DERIVATIVE	A. *Infinitive*	(a) *Triliteral Verb*	several patterns, e.g., cica|cat, cucuwc.	'dira|sat, "study" 'nuzuwl', "arrival"
		(b) *Quadriliteral Verb*	cacocac →→ cicoca|c, cacocacat.	'ziloza|l, zalozalat' "earthquake"
		(c) *Mono-Augmented Verb*	:acocac →→ :icoca|c. caccac →→ tacociyc. ca|cac →→ cica|c, muca|cacat.	':ikoba|r', "honouring" "taqodiyr', "estimate" 'qita|l, muqa|talat' "fighting"
		(d) *Poly-Augmented Verb*	:icotacac →→ :icotica|c. tacaccac →→ tacaccuc. :icocanocac →→ :icocinoca|c. :icocacacc →→ :icocicoca|c. :icocacc →→ :icocica|c. :inocacac →→ :inocica|c. :isotacocac →→ :isoticoca|c. tacacocac →→ tacacocuc. :icocawocac →→ :icociyca|c.	':ijotima|ɛ', "meeting" 'takassur', "breaking" ':iforinoqa|ɛ', "explosion" ':iqosviɛora|r', "shivering" ':ih-omira|r', "reddening" ':inot;ila|q', "departure" ':isotiqoba|l', "reception" 'tadah-oruj', "rolling" ':ih-odiyda|b', "curvature"
	B. *Qualificative*			
	1) *Active Verbal*	(a) *Triliteral Verb* (b) *Imperfect Non-Triliteral Verb*	→→ ca|cic. prefix →→ 'mu', penultimate vowel →→ <i>.	'na|zil', "arriving" 'mudarris', "teaching"
	2) *Passive Verbal*	(a) *Triliteral Verb* (b) *Imperfect Non-Triliteral Verb*	→→ macocuwc. prefix →→ 'mu', penultimate vowel →→ <a>.	'madoruws', "studied" 'mudarras', "taught"
	3) *Assimilated Adjective*	*Intransitive Triliteral Verb*	cacic →→ cacic, :acocac, cacoca|n. cacuc →→ cacac, caciyc, cicoc, cuca|c, caca|c, cucoc.	'jad-il', "happy" 'h-asan', "good" 'kabiyr', "big"
	4) *Comparative*	*Triliteral Verb*	→→ :acocac.	':as;ogar', "smaller"
	5) *Exaggerative*	*Triliteral Verb*	several patterns, e.g., cacca|c, cacuwc.	'd;ah-h-a|k, d;ah-uwk' "given to laughter"
	C. *Derived Substantive*	(a) *Triliteral Verb* (b) *Non-Triliteral Verb*	→→ macocac, macocic. Passive Verbal pattern.	'makotab', "desk" 'mujotamaɛ', "venue"
	1) *Tlocative*			
	2) *Instrumental*	*Triliteral Verb*	→→ micocac, micocacat, micoca|c.	'miborad', "sharpener" 'mifota|h-', "key"
	3) *Numeral*	*Triliteral Verb*	several patterns, e.g., caca|c, cacac, cacoc.	't-ala|t-', "three"

TABLE 29: Nominal Types and Derivational Processes in M.S.A.

We note, in Table 29, that Derivative Nouns include Qualificatives such as '*:isom :alofa/ɛil*', and '*:isom :alomafoɛuwl*', literally and respectively, "Noun of the Agent" and "Noun of the Patient" but more usually translated as "*Present or Active Participle*" and "*Past or Passive Participle*" respectively. The trouble with such terms as "*Noun*" for ':isom', "*Agent*" for 'fa|ɛil', "*Patient*" for 'mafoɛuwl', and "*Participle*" for the last two, is the confusion they give rise to because of their connotations in Western European linguistics. For instance, CRYSTAL, 1980: 258, defines a *participle* as "a word derived from a VERB and used as an adjective", but he notes, loc. cit., that "in LINGUISTICS the term [*participle*] is generally restricted to the non-FINITE forms of Verbs other than the INFINITIVE". However, he rightly objects, loc. cit., to the use of the labels *present* and *past* because of their "Latinate associations of time" which are "inapplicable" in certain English examples. However, in Arabic, these associations of time are indeed an integral part of the definition of "Noun of the Agent" and "Noun of the Patient", since they always occur with a TENSE value: imperfect, if they are used on their own, or a TENSE value modified by an Auxiliary, if they are used with one (see also NIƐMA, 1973: Vol. II, 40–41 and 45). Another associated problem is that, in Western European linguistics, the definitions of the terms *Noun* and *Participle* are mutually exclusive, whereas, in Arabic, the two classes display differences, and also share similarities. Hence, we suggest the cover term *Nominal*—instead of *Noun*—to include Primitives and Derivatives, and the term *Verbal* instead of the Arabic '*:isom fa/ɛil*' and '*:isom mafoɛuwl*' (commonly rendered, as we have seen, by the term *Participle*). We shall henceforth call an *ACTIVE Verbal*, *Verbal1* and a *Passive Verbal*, *Verbal2*, and use the term *Noun* to refer to Primitives only.

I.1.1 The Arabic Verbal

In M.S.A., the *Verbal* CATEGORY is a hybrid one in that it is an amalgam of Verb and Noun features. Thus, it has similarities and differences with both the Verb and the Noun.

I.1.1.1 Similarities between Verbals and Verbs

Looking at the similarities between Verbals and Verbs, we find that they share features of VOICE, TENSE, TRANSITIVITY, GOVERNMENT, syntactic function, enclitic Pronoun suffixation, and derivation. Like Verbs, Verbals may display active and passive contrasts of VOICE by the use of morphological pattern structure. *Activization* involves either the infixation of <|> in the perfect pattern of a Triliteral Verb, or the prefixation of 'mu' with vocalic changes to the imperfect pattern of a non-Triliteral Verb. *Passivization* involves the use of the pattern 'macocuwc', when processing a Triliteral Verb; and the prefixation of 'mu' with vocalic changes, when processing the imperfect pattern of a non-Triliteral Verb. This is illustrated in Table 29.

Verbals also display TENSE contrasts; and when used without Auxiliaries, they have an imperfect TENSE aspect. However, in M.S.A., Verbals used with Auxiliaries, such as 'ka|na', "was", can have a perfect TENSE aspect, as in ':alddarosu ka|na makotuwba+', "the lesson was written".

The TRANSITIVITY of Verbals is similar to that of Verbs. There is a general two-way distinction between transitive and intransitive Verbals. Within these two categories, further distinctions are possible, based on the syntactic nature of the Objects or Complements required. Objects and Complements generally divide into Nominal and Prepositional. Membership in one or the other of the classes of TRANSITIVITY depends on whether the root is intensive or causative. As is the case for Verbs, causativization PROMOTes Verbals to a higher TRANSITIVITY rank while passivization DEMOTes them to a lower one. However, the TRANSITIVITY value: *ptransitive* does not occur with Verbals, since they do not have an impersonal use as seen for Verbs in Table 2. Further, Verbals differ with Verbs in having one new TRANSITIVITY value, which is *mxtransitive*, and which is defined below. This new value is ASSIGNed because of the ambiguity of Verbals such as 'h-a|sibu+', "thinking-he", or "counting-he", with two possible interpretations, and therefore two possible TRANSITIVITY values: *monotransitive* or *xtransitive*. Table 30 below shows TRANSITIVITY contrasts of the Verbal in M.S.A.

In syntactic terms, Verbals share with Verbs two main characteristics: GOVERNMENT and syntactic function. Both Verbs and Verbals govern Subjects in the nominative CASE, and Objects in the accusative CASE. Thus, their role in the sentence is to predicate over other arguments. Hence, it is the case that, for both Verbs and Verbals, suffixation of enclitic Pronouns is subordinated to the TRANSITIVITY of the Verb or Verbal itself.

Finally, from a derivational point of view, Verbals themselves are Derivatives, in the sense of being derived from roots with the same meaning as that of Verbs. Thus, from 'drs', "to study", we can derive the Verb 'darasa', "he studied", and the Active Verbal 'da|ris', "studying", and the Passive Verbal 'madoruws', "studied". As hinted above, the derivation in question involves the application of morphological rules consisting of changes to vocalic structure, and/or affixation to the Verb pattern. A summary of contrasts between Verbs, Verbals, and Nouns in M.S.A. is given in Table 31 below. Note that in this table, VOICE is a property which characterizes the morphological level, and *PERSON* refers to the property *PERSON* with the value third which is ASSIGNed to Verbals and Nouns by default.

TRANSITIVITY		SYNTACTIC REQUIREMENT	
		DEFINITION	EXAMPLE
Intransitive		requires no Objects.	'ja|lisu+' "sitting-he"
TRANSITIVE	*Monotransitive*	requires one Nominal Object.	'da|risu+' "studying-he"
	Ditransitive	requires two Nominal Objects.	'ma|nih-u+' "granting-he"
	Xtransitive	requires one Nominal Object; and one Nominal or Prepositional Complement.	'd/a|nnu+' "considering-he"
	Cotransitive	requires one Nominal or Prepositional Complement.	'mad/onuwnu+' "considered-he"
	Nctransitive	requires one optional Nominal or Prepositional Complement.	'mah-osuwbu+' "thought-he/ counted-he"
	Mxtransitive	requires one Nominal Object; and one optional Nominal or Prepositional Complement.	'h-a|sibu+' "thinking-he/ counting-he"
	Ptransitive1	requires one Prepositional Object.	'na|zilu+' "arriving-he"
	Ptransitive2	requires one Nominal and one Prepositional Objects.	'ma|niɛu+' "preventing-he"

TABLE 30: TRANSITIVITY of the Verbal in M.S.A.

I.1.1.2 Differences and Similarities between Verbals and Nouns

The very similarities between Verbs and Verbals are those that distinguish Verbals from Nouns in M.S.A. Thus, Nouns do not have features for VOICE, TENSE, TRANSITIVITY, or GOVERNMENT. Unlike Verbs and Verbals, Nouns do not govern Subjects or Objects, and do not predicate. They are used in an argument—or non-predicative—syntactic role. Suffixation of clitic Pronouns to Nouns is a process that is free of any of the TRANSITIVITY constraints governing the suffixation of such Pronouns to Verbs and Verbals. Further, Nouns, or Primitive Substantives, have unpredictable derivation, in that their patterning is idiosyncratic and irregular. The Substantives, including Common and Proper Nouns, are usually divided further into human and non-human Nouns. They are unlike Infinitives in that an Infinitive has a stem that can be used to derive a Verb as described above. Moreover, Infinitives are also characterized by being devoid of TENSE, VOICE, TRANSITIVITY, or GOVERNMENT. The distinction

CRITERION	VERB	VERBAL	NOUN
VOICE	YES		NO
TENSE	YES		NO
TRANSITIVITY	YES		NO
GOVERNMENT	REGENT		NON-REGENT
SYNTACTIC FUNCTION	PREDICATOR		ARGUMENT
CLITIC SUFFIX	TRANSITIVITY DEPENDENT		FREE
DERIVATION	DERIVATIVE		PRIMITIVE
MOOD	YES	NO	
CASE	NO	YES	
DEFINITENESS	NO	YES	
HUMANITY	NO	YES	
SUFFIX VALUE	NGP	NGC & PERSON	

TABLE 31: A Comparison of Verbs, Verbals, and Nouns in M.S.A.

of Common and Proper Nouns is morphologically relevant mainly because of the differences between them, described in § II.1.6.1 below, in *definitization*. The relevance of the distinction of human and non-human Nouns is explained in Chapter 12, § I.2.

However, we can see from Table 31 that Verbals are different from Verbs, and similar to Nouns in many respects. Thus, neither Verbals nor Nouns carry MOOD values. Instead, they both carry CASE values. While both Verbs and Nominals display NUMBER, GENDER, and PERSON values, the affixes used for such contrasts are different for Verbs—which use conjugation affixes as described in Chapter 4—and for Verbals and Nouns, both of which use Nominal-inflection suffixes. Verbals and Nouns share another common feature which is DEFINITENESS. Both types are definitizable with the prefixation of the Determiner ':alo', "the"; while Verbs are never definitized (except in some variants of Classical Arabic, of which ALSAYYID, 1982: Vol. I, 30, gives an example, but which do not concern us here). Further, both Verbal and non-Verbal Nominals can occur in genitive Nominal groups either as *Annexed* (*'mud;a/f'*) or as *Dependent* (*'mud;a/f :ilayoh'*) arguments, and in other Nominal groups as heads. Finally, both Verbals and Nominals can be classified on the basis of a HUMANITY feature, into human and non-human; while Verbs do not carry any values for HUMANITY.

We can conclude from the comparison between Verbs, Verbals, and Nouns, that Verbals occupy an intermediate position between Verbs and Nouns, sharing some features with the former and others with the latter.

I.1.2 Non-Verbals in M.S.A.

In Table 29 above, we noted two different types of Substantive: those that are Primitive, and those that are Derived. Primitive Substantives can be subdivided into Common and Proper Nouns and into Nouns denoting human and non-human entities. Derived Substantives are of three types: *Tlocative*, *Instrumental*, and *Numeral*. A *locative* is defined by CRYSTAL, 1980: 214, as that which "refers to the FORM taken by a WORD, usually a NOUN or PRONOUN, when it typically expresses the idea of location of an action". Here, we use the term *Tlocative*, or *Temporal-Locative*, where *Temporal* is used as before with the meaning of "pragmatic-time reference", and *Locative* is used with the meaning of "location of an object or an action". Thus, we use the term *Tlocative*, to refer to what is traditionally called *Nouns of Time and Place*, since these have the same type of pattern, e.g., 'macocac', 'macocic', etc; and indeed since it is very often the case that a word form is used with both "temporal" and "locational" meanings as in 'magorib', "the (place of) sunset".

The Tlocative, Instrumental, and Numeral Derived Substantives are set apart from Primitives by their derivational processes: they are derived according to regular processes of patterning. These Substantives are also semantically different. While Primitive Substantives represent entities—or items—Derived Substantives represent *situational variables*, as described above. Numerals, however, can be seen as *Quantifiers* which specify the quantity of items and objects.

However, from a syntactic point of view, all Primitive Substantives, Tlocatives, and Nouns of Instrument are used as *Non-Predicative Arguments*. Numerals are mostly used as Modifiers. This is a feature that they share with another type of Qualificative—listed in Table 29—which is *Assimilated Adjectives*. These are so called because their form is generally thought to be assimilated to that of ':isom :alofa|ɛil'. Thus, they are called '*s;ifa/t musvabbahat bi:isom :alofa/ɛil*', literally, "Adjectives which are compared to the Active Verbal". In M.S.A., these Adjectives are used in the pre- and post-modification of other Nominals, as in 'ɛamalu+ h-asanu+', "a good job".

I.1.3 The Common Features of Verbal and Non-Verbal Nominals in M.S.A.

I.1.3.1 Graphological Nominal Structure

Although Verbals have roots which are more similar to Verb roots, in meaning, TRANSITIVITY, and augmentation, both Verbal and non-Verbal roots can be classified on the basis of graphological structure, i.e., the graphemic nature of the constituent radicals of the root. For convenience of exposition, we shall refer to Verb and Verbal roots as *roots* and to Nominal

roots as *stems*. We shall understand by *stem* the set of non-vocalic graphemes which occur at the edges of syllables in the Nominal. Further, while the graphological analysis of V-Roots had to study all radicals—initial, medial, and final—the point of focus in the study of the graphological structure of the Nominal is the final radical. This is because the conjugation of Verbs with Subject Pronouns (as explained in Ch. 4) can lead to changes in any of the radicals of the Verb, especially if it happens to be Defective, whereas the inflection of Nominals for NUMBER, GENDER, and CASE, leads to changes mostly in the final radical of the Nominal. Hence, the Nominal can be subdivided into two types, according to which type of final radical it contains. If the final radical is <:> preceded by 'a|' or a semivowel, the Nominal is said to be *Defective*. If this radical is a consonant other than <:> preceded by 'a|', the Nominal is said to be *Sound*. Table 32 below gives a classification of Nominals in M.S.A. on the basis of the graphological nature of their radicals.

GRAPHOLOGICAL NOMINAL STRUCTURE			
TYPE		DEFINITION	EXAMPLE
DEFECTIVE	*Extended*	a Nominal that is extended with the addition of the suffix: 'a|:'.	'zaroqa|:' "blue"
	Abbreviated	a Nominal whose final radical is <y> or <|> of prolongation.	'mabonay' "a building"
	Weak	a Nominal that ends in 'iy'.	'ba|niy' "builder(s)"
Sound		a non-Extended Nominal, that has a non-semivocalic final radical.	'qalam' "a pen"

TABLE 32: Graphological Structure of the Nominal in M.S.A.

I.1.3.2 Morphological Pattern Structure

Besides the patterning involved in the generation of Primitive and Derivative Nominals, there are other derivational processes used in their generation. These processes depend on the desired NUMBER and GENDER for the given Nominal form. Thus, we can have masculine and feminine Nominals which may be singular, dual, or plural. In M.S.A., all singular-masculine and Broken-Plural Nominals are generated by the use of derivational patterns; while dual and Regular-Plural (RP) Nominals are generated by the addition of NGC suffixes. Feminine Nominals can be generated by either process. HAMMOND, 1988: 253 et passim, argues that Arabic Broken Plurals should be derived phonologically from their singular counterparts. However, he

deals only with a subset of such Nominals (p. 253). In a few circumspect remarks (p. 262, fn. 14, p. 264, and p. 267, fn. 16), he concedes that there are "a number of counterexamples" to his analysis, but dismisses them on the grounds that they are "unpredictable" and "different" from his chosen cases; and so he proposes, ibid. 262, fn. 14, a lexical "solution" whereby all forms are listed, "marked for whether they take a broken plural or not", and then the "singulars [for unpredictable plurals] are marked not to undergo [the regular derivation] rule". Such arguments amount to saying that the derivation of Arabic Broken Plurals is predictable (and thus can be treated by rule, somewhere in the grammar, e.g., the phonology, for HAMMOND), but that it is also unpredictable (and is thus treated in the lexicon). A similar analysis is advanced by ANDERSON, 1982. This type of approach implies extremely difficult organizational problems relating to the need for lexical marking as well as for further blocking rules and constraints to prevent multiple application of the same rule to material that has already been processed in the lexicon.

In § I.2 of this Chapter, we discuss further distinctions of Nominal CATEGORIES, and in Table 33 below we summarize the distribution of the Nominal processes for the generation of NUMBER and GENDER categories.

Note that all Nominals other than Personal Pronouns have default PERSON values. Personal Pronouns include Subject Pronouns with feature dimensions, and feature values as listed in Tables 6 and 7.

I.1.3.3 Declinability and CASE Inflection

Another feature common to all Nominals is their inflection for CASE. Generally, all Nominals are inflected for one of three CASES: *nominative, accusative*, or *oblique*. We use the term *oblique*, here, as a translation for the Arabic '*majoruwr*', and with a meaning based on a modified definition of this term by MURRAY et al., 1989: Vol. X, 651. In particular, we mean by an *oblique* CASE any CASE other than the nominative and the accusative. Therefore, the term *oblique* will cover both the *genitive* CASE, which is taken by a Nominal that is governed by another Nominal, and the *prepositional* CASE, which is taken by a Nominal that is governed by a Preposition. In Arabic, both the genitive and the prepositional CASES are indicated by one CASE MARK called '*kasorat*', "oblique (MARK)", and hence the use of the term *oblique* for both.

However, the distribution of the three CASES among Arabic Nominals is not always on a clear-cut basis. Thus, not all Nominals are capable of showing the versatility of having a distinct CASE MARK for each of the three CASES. Indeed, certain Nominals, such as *Abbreviated* Nominals are totally indeclinable for CASE: they have no distinctive surface CASE endings, or CASE MARKS. Hence, they can be seen as *neutral* for CASE. Other indeclinable Nominals

NUMBER	GENDER	DEFINITION	PATTERN & PROCESS	EXAMPLE
Singular	*Masculine*	indicates 1	cica|c, cacac, caca|c, PNs, etc.	'kita|bu+' "a book"
	Feminine		cucc, macocac + suf, PNs, etc.	':ummu+' "a mother"
Dual	*Masculine*	indicates 2	masculine singular + suf.	'kita|ba|ni' "two books"
	Feminine		feminine singular + suf.	':umma|ni' "two mothers
Regular Plural	*Masculine*	indicates more than three.	masculine singular + suf.	'ka|tibuwna' "writers"
	Feminine		feminine singular + suf.	':umma|tu+' "mothers"
Broken Plural of Paucity	*Masculine*	indicates three to ten.	specified patterns, e.g., cucca|c.	'kutta|bu+' "writers"
	Feminine		specified patterns, e.g., :acoca|c.	':aqoda|mu+' "feet"
Broken Plural of Abundance	*Masculine*	indicates more than three.	specified patterns, e.g., cica|c.	'jiba|lu+' "mountains"
	Feminine		specified patterns, e.g., caca|:ic.	'kata|:ibu+' "regiments"

TABLE 33: Morphological Pattern Structure

have just one invariable CASE for which they are fixed. For instance, Subject Pronouns are always fixed for the nominative.

Other Nominals are generally declinable for the three CASES. However, these have varying degrees of declinability. Singular and certain BP Nominals are *fully declinable of Type 1*: they have distinctive CASE MARKS for all three CASES. Dual, RP Nominals, and certain Demonstrative Pronouns are *fully declinable of Type 2*: they have a distinctive CASE MARK for the nominative CASE, and only one other mark for both the accusative and oblique CASES. Other declinable Nominals, are only partially declinable, or *diptote*. *Diptotes of Type 1* are not allowed to have oblique vowel marks; nor are they allowed to have 'tanowiyn', i.e., full vowel marks. Thus, both '*jawa|hiri' and '*jawa|hiri+', "*diamonds", are not allowed; and the accusative mark will then cover also the oblique CASE. *Diptotes of Type 2*—which are Feminine RPs—are not allowed to have any accusative vowel marks. Thus, '*da|risa|ta' and '*da|risa|ta+', "*students", are not allowed; and the oblique mark will then also cover the accusative CASE. Finally, *Diptotes of Type 3*—which are Wk.Ns—are not allowed to have any

DECLINABILITY		CMK	'TANOWIYN'	CATEGORY	EXAMPLE
D E C L I N A B L E	*Fully Declinable of Type 1*	nmm	allowed	singular, BP & some patterns, e.g., cica|c.	'kita|bu+' "a book"
		acm	allowed		'jiba|la+' "mountains"
		obm	allowed		'kutta|bi+' "writers"
	Fully Declinable of Type 2	nmm	N/A.	dual, RPs, & some DPs.	'ka|tiba|ni' "two writers"
		acm-obm	N/A.		'ka|tibuwna' "writers"
	Diptote of Type 1	nmm	prohibited	some BPs & PNs; & Ex.Ns.	'mana|zilu' "houses"
		acm	prohibited		's;afora|:a' "yellow"
		*obm	N/A.		
	Diptote of Type 2	nmm	allowed	FRPs.	'dira|sa|tu+' "studies"
		*acm	N/A.		
		obm	allowed		'dira|sa|ti+' "studies"
	Diptote of Type 3	*nmm	N/A.	Wk.Ns.	
		acm	allowed		'ba|niya+' "a builder"
		obm	allowed		'ba|ni+' "a builder"
Indeclinable		nmm	N/A.	SPs.	':anota', "you"
		acm	N/A.	Free CPs.	':iyya|ka', "you"
		obm	N/A.	GPs.	'hu', "his"
		NEUT	N/A.	some DPs, RPs; & Ab.Ns.	'had-a|', "this" 'mabonay' "a building"

TABLE 34: Declinability and CASE Inflection

PRONOUN		NGC VALUES		
		NUMBER	GENDER	CASE
1	'had-a|' "this"	singular	masculine	neutral
2	'had-ihi' "this"		feminine	
3	'had-a|ni' "these two"	dual	masculine	nominative
4	'had-ayoni' "these two"			accusative-oblique
5	'ha|ta|ni' "these two"		feminine	nominative
6	'ha|tayoni' "these two"			accusative-oblique
7	'ha:ula|:i' "these"	plural	masculine-feminine	neutral

TABLE 35: Demonstrative Pronouns in M.S.A.

ominative vowel marks. Thus, '*ba|niyu' and '*ba|niyu+', "*a builder", are not allowed; and he oblique mark will then also cover the nominative CASE.

These restrictions apply only when these Nominals are in the indefinite form, and are largely ependent on NUMBER and GENDER categories, as illustrated in Table 34 above, which ummarizes this discussion on the declinability and CASE inflection of Nominals. Table 35 bove shows NGC values for Demonstrative Pronouns in M.S.A.

.1.3.4 FLEXION and DEFINITENESS

Ve discussed, in § I.1.3.3 above, the topic of CASE inflections, and we hinted that each CASE i indicated by CASE MARKS with various degrees of syncretism. Indeed, for fully declinable lominals, each CASE is indicated by one of two different types of CASE MARKS. These can ither be vowel marks—which carry only CASE, but no NUMBER/GENDER values—or they an be longer (*NGC*) suffixes, which carry both CASE and NUMBER/GENDER values. Each owel mark can be either a *single* vowel as in 'kita|bu', "(the) book of"; or a *full* vowel as i 'kita|bu+', "a book". Each NGC suffix can have either a *full*—or complete—form as in)anuwna', "sons"; or a *reduced*—or partial—form as in 'banuw', "(the) sons of".

We shall refer to this aspect of CASE inflection as the *FLEXION* feature, and to the value of ach FLEXION as the CASE feature. Thus, 'kita|bu' and 'kita|bu+' above have two different

FLEXIONS, namely single and full vowels, but one CASE value, namely nominative. Further, we will refer to a complete NGC suffix as a *full suffix* and to a partial NGC suffix as a *reduced suffix*. The distinction between FLEXION and CASE is important in two respects. First, with respect to DEFINITENESS, and second with respect to categorization.

There is in fact a strong correlation between FLEXION and DEFINITENESS, in that definite Nominals that are morphologically definitized by the prefixation of the Determiner ':alo', "the", never have full vowel marks. Thus, '*:alokita|bu+', "*the a book", is not allowed. Similarly, such definite Nominals rarely have reduced NGC suffixes. The only instance in which this occurs, is in Numerals such as ':alo:it-ona|', "the two", ':alomi:ata|', "the two hundred", and so on. Other definite Nominals, i.e., those that are exophorically—and not morphologically—definitized, have one frozen, or invariable, FLEXION. These include Subject, Demonstrative, and Relative Pronouns. However, in M.S.A., Proper Nouns have only single/or full vowel marks. In C.A., Proper Nouns are also dualized and pluralized, as well as definitized, as in 'zayoda|ni', "two Zayds", ':alzzayoduwna', "the many Zayds", and ':alzzayodu', "the Zayd". A pertinent remark here is ANGHELESCU's, 1974: 49, objection to this type of definitization. She notes that

> one cannot conceive of definitization in the case of Proper Nouns, especially when we think of definitization as an operation (otherwise we will have difficulties in explaining nunation [i.e., full vowel marks] in Proper Nouns).

Indefinite Nominals have more freedom in NGC suffixation: they can have either single/or full vowels; and either reduced/or full NGC suffixes. Table 36 below shows the correlation between FLEXION and DEFINITENESS in M.S.A.

I.1.3.5 FLEXION, Definitization, and Categorization

We hinted, in § I.1.3.4 above, that the distinction of FLEXION types is important with respect to the categorization of Nominals. Indeed, there is a strong correlation between FLEXION and categorization for Nominals:

a. The strongest and most obvious correlation is that, in the case of indefinite Nominals with single vowel mark or with reduced NGC suffix, the only possible syntactic role they can have is as Modifiers. In this class, we have distinguished two types of Modifiers. The first, labelled *MD*, is an Annexed Modifier in a genitive construction—also called *Annexation* (*':id;a/fat'*)—which implies that the second element of the construction, or *Dependent*, must have oblique CASE. The second type, labelled *MM*, is a Numeral Modifier, which is slightly different: it can occur either in a genitive construction—which requires of the Dependent that it be in the oblique CASE—or in a *Specification* construction, which requires of the Specifier that it be in the

TYPE	DETERMINER	FLEXION	CATEGORY	EXAMPLE
Indefinite	bare of Determiner.	svm/fvm dnf/nnf	Nominals.	'rajulu', "man" 'rajulu+', "a man" 'rajula|ni', "two men"
Definite	with attached Determiner, (morphological).	svm dnf/nnf	Nominals.	':alrrajulu', "the man" ':alo:it-ona|', "the two" ':alrrajula|ni' "the two men"
	bare of Determiner, (exophoric).	svm/fvm	PNs.	'zayodu, zayodu+', "Zayd"
		frozen	Pronouns: SPs, DPs, RPs, QPs, etc.	':anota', "you" 'had-a|', "this" ':allad-iy', "who" 'ma|', "what" 'kayofa', "how"

TABLE 36: Correlation between FLEXION and DEFINITENESS

accusative CASE. This depends entirely on the NUMERICAL VALUE of the Numeral. We have called this role, for want of a better term, a *quantification* role. (For further explanation and examples of these roles, cf. our discussion in Ch. 12, § I.)

b. Indefinite Nominals with full vowel mark or with full NGC suffix divide into four types:

First, indefinite Verbals—which are of two types, active and passive, respectively labelled *L1* and *L2*. Syntactically, they govern Subjects in the nominative CASE and Objects and Complements in the accusative CASE. Hence, they have a TRANSITIVITY feature specifying how many Objects are required. Thus, they act as *regents* and as *predicators*. The *predicative* role is constrained syntactically and requires a topicalized, interrogative, or negative sentence which is in the present TENSE. However, such conditions can be specified in the syntax. There are also morphological constraints on this role: BPs—as opposed to RPs—have only a non-predicative role.

Second, indefinite Substantives, Tlocatives, and Instrumentals—although different in derivation—form one syntactic CATEGORY which is that of Noun and which we have labelled *NN*. Syntactically, Nouns typically occur as heads of Nominal groups, but may also occur in genitive constructions as dependent elements.

Third, Adjectives—which we have labelled *AAs*—and which occur as *attributive* in Nominal groups and as predicative in sentences.

Fourth, Numerals—which we have labelled *MMs*—and which occur also as attributive in Nominal groups, and as predicative in sentences. However, the syntactic relationships—especially those relating to agreement—of Adjectives and of Numerals to their Nominal groups or sentences are entirely distinct. Again, this is determined by the NUMERICAL VALUE of the Numeral, hence the distinction in categorization. However, it is evident that the correlation between FLEXION and categorization is opaque with respect to Numerals, since NUMERICAL VALUE is vital in determining their syntactic roles and relationships, whereas their different possible types of FLEXION do not suggest discrete Numeral CATEGORIES. Hence, our common label *MM*, to cover indefinite Numerals that are of single vowel or of full vowel marks, and with full or reduced NGC suffix.

c. The third group of Nominals is the definite group, which includes indeclinable Nominals such as Pronouns, and also Substantives, Tlocatives, Instrumentals, and Proper Nouns. These have the same syntactic roles as their indefinite counterparts, but here are labelled *DNs*. This group also includes Verbals, which when definitized occur in (non-predicative) argument positions. Thus, their categorization is dynamic and depends on DEFINITENESS. In C.A., definite Verbals are also used as predicative elements. Definite Adjectives and Numerals can be used in the same way as their indefinite counterparts, but occur mostly in attributive positions. However, here again the agreement rules are different for each CATEGORY. We have labelled these, respectively, *DAs* and *DMs*.

The distinction of such CATEGORIES of Nominal is of vital importance in the anticipation of syntactic work on areas such as agreement and GOVERNMENT, and serves to improve the optimization of the MA, hence the above discussion. Table 37 below summarizes the correlation between FLEXION types, definitization, and categorization in M.S.A.

I.1.4 The Arabic Nominal System

To summarize this introductory section on Nominals, we can say that the Arabic Nominal is a complex system of overlapping levels of contrast. The derivational and inflectional levels enter into morphological and graphological interplay to generate surface realizations of underlying stem and pattern forms, which carry information about TRANSITIVITY, VOICE, and TENSE features—in the case of Verbal Nominals—and about NUMBER, GENDER, CASE, FLEXION, and DEFINITENESS features, in the case of all Nominals.

In addition, the stems of Nominals carry information about the HUMANITY or non-HUMANITY of the entity they refer to, about the plurality of that entity, and—in the case of Numerals—about the NUMERICAL VALUE they can impart to a given Nominal form. Further, each Nominal form is characterized by radical type, pattern structure, and declinability. The overall system makes up the Nominal CATEGORY, although we have distinguished

TYPE	PATTERN TYPE	FLEXION	CATEGORY	EXAMPLE
	Substantive	fvm/nnf	NN	'kita|bu+', "a book"
	Numeral	fvm/nnf	MM	'sittu+', "six"
	Adjective	fvm/nnf	AA	'kabiyru+', "big"
	Verbal1	fvm/nnf	L1	'mudarrisu+', "teaching"
I	*Verbal2*	fvm/nnf	L2	'madoruwsu+', "studied"
N	*Tlocative*	fvm	NN	'manozilu+', " a house"
D	*Instrumental*	fvm/nnf	NN	'mifota|h-u+', "a key"
E	*Substantive*	svm/dnf	MD	'kita|bu',
F				"(the) book (of)"
I	*Adjective*	svm/dnf	MD	'kabiyra|', "(of) big"
N	*Verbal1*	svm/dnf	MD	'mudarrisu',
I				"(the) teacher (of)"
T	*Verbal2*	svm/dnf	MD	'madoruwsu',
E				"(of) studied..."
	Tlocative	svm	MD	'manozilu',
				"(the) house (of)"
	Instrumental	svm/dnf	MD	'mifota|h-u',
				"(the) key (of)"
	Numeral	svm/dnf	MM	'sittu', "six (of)"
	Substantive	svm/nnf	DN	':alokita|bu', "the book"
	Adjective	svm/nnf	DA	':alokabiyru', "the big"
D	*Verbal1*	svm/nnf	DN	':alomudarrisu'
E				"the teacher"
F	*Verbal2*	svm/nnf	DA	':alomadoruwsu'
I				"the studied"
N	*Tlocative*	svm	DN	':alomanozilu', "the house'
I	*Instrumental*	svm/nnf	DN	':alomifota|h-u', " the key"
T	*Numeral*	svm/dnf	DM	':alssittu', "the six"
E	*PNs*	svm/fvm	DN	'zayodu+', "Zayd"
	SPs	frozen	DN	':ana|', "I"
	DPs	frozen	DN	'had-a|', "this"

TABLE 37: Correlation between FLEXION, Definitization, and Categorization

several subcategories inside it, mainly Verbal and non-Verbal. The prediction of syntactic

slots and positions for each subcategory together with the distinction of HUMANITY, NUMERICAL, DEFINITENESS, FLEXION, and declinability types will play a vital role in the triggering of syntactic rules of agreement and GOVERNMENT.

To give an example, the Nominal form 'banuwna', "sons", indicates 3 (M) P, nmm, ndf, nnf, hm, rmp, NN. It has a Df.Wk. stem 'bn', "child", and abstract pattern 'cac', and a full NGC suffix 'uwna'. In addition, the discontinuous stem 'bn' has the semantic content 'child'; and the whole structure denotes the meaning 'sons'. Table 38 below shows a multidimensional representation which illustrates the complexity of the Arabic Nominal system.

I.2 SCOPE OF THE N-ANALYSIS

The present analysis attempts to select—for study—a set of Arabic Nominals that is representative of most of the classes of Nominal described so far. Similarly to the selection of Verb classes, here again, the overriding concern was selection and representativity rather than lexical exhaustiveness, in the sense of studying a large number of lexical items. Owing to time and space constraints, such exhaustiveness was not possible, and the total number of Nominal stems was thus limited to 149, 148 of which are the same as the V-Roots used in Chapter 4, the other stem is 'stt', "six", which does not have any derivative Verb forms, and can only be Nominal. The patterns used with these stems, in order to derive *Nominal Forms* (*NFs*), are obviously different from those used in the derivation of Verb forms; and different for each CATEGORY of Nominal. A complete list of these stems, their derivational CATEGORIES, and patterns is given in Appendix A, § B.

These CATEGORIES cover representative samples from all the Primitive Nominals—including (human and non-human) Substantives—and Derivative Nominals, including Verbals, Adjectives, and some Infinitives, as well as Derived Substantives, such as Tlocatives ("Nouns of Time and Place"), Instrumentals, and Numerals. They also include examples from the Sound and Defective (Abbreviated, Extended, and Weak) classes. They include Common and Proper Noun types of Nominal and Nominal substitutes, such as Subject and Demonstrative Pronouns.

Also included are:

- all (fully and partially) declinable Nominal and indeclinable types in all CASES—including neutral—in all FLEXIONS, and in all NUMBER and GENDER categories, including masculine, feminine, singular, dual, RP, and BP categories,
- indefinite and, morphologically and exophorically, definite Nominal classes, and

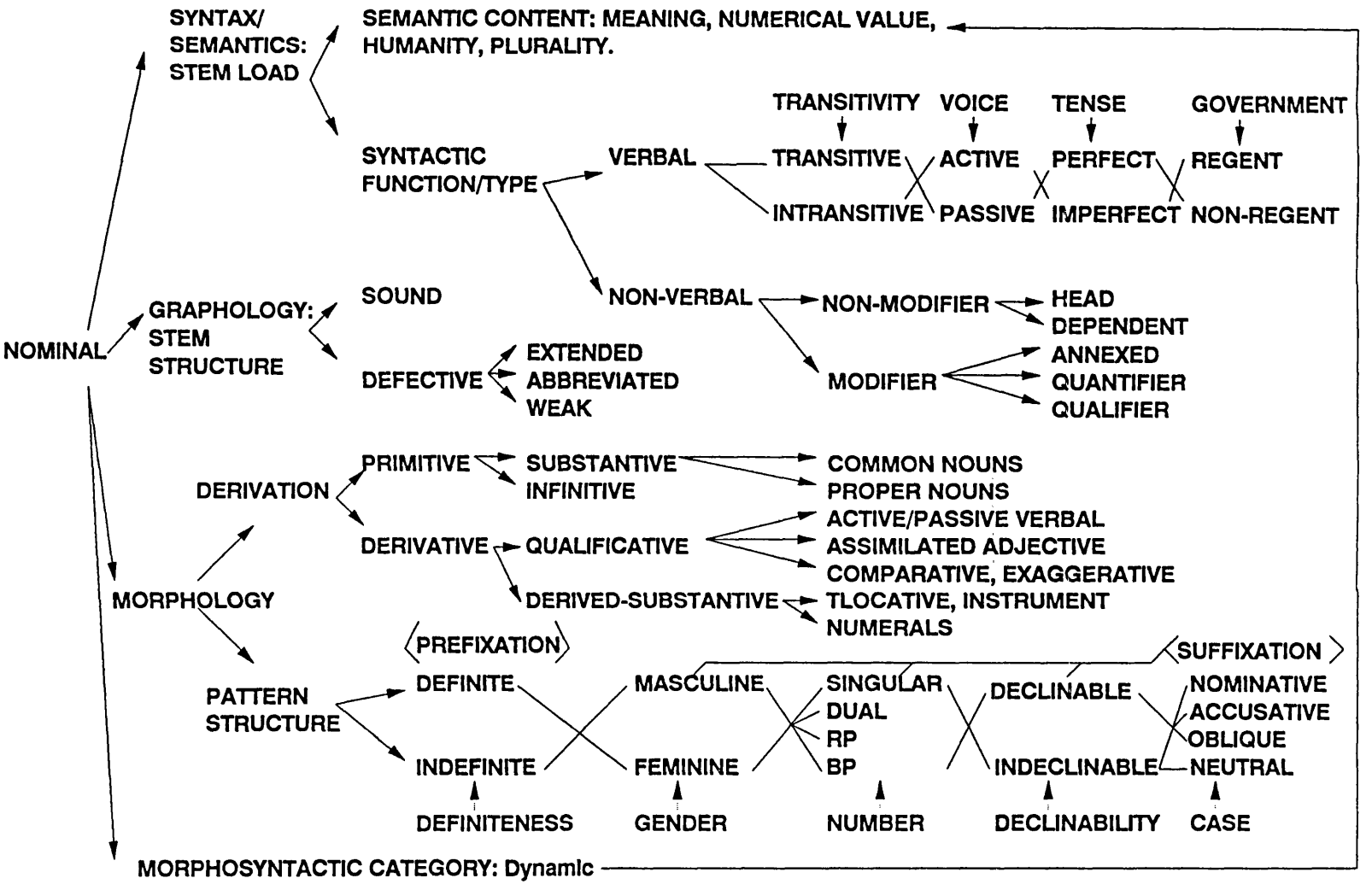


TABLE (38): The Arabic Nominal System

- Verbal—including active and passive CATEGORIES—with all the different TRANSITIVITY values listed in Table 30, and non-Verbal CATEGORIES—including Modifiers, which are Quantifiers or Numerals, Qualifiers or Adjectives—and non-Modifiers.

However, as noted earlier, we have not distinguished Instrumentals and Nouns of Time and Place—or Tlocatives—from Primitive Substantives. We have put these together in one morphosyntactic CATEGORY. Neither did we distinguish plurals of paucity and abundance, nor *Hyper* or *Super* plurals. Here, all these groups have the same derivational process, and the distinction is purely semantic, if it is valid at all. For instance YAEQUWB, 1986: 242, objects to such distinctions, noting that "what has become clear to us from an objective reading of the language reality, is that all forms of the plural are used with both the meanings of *paucity*, and of *abundance*, depending on their context"; while ABU ALMAKAARIM, n.d. 170–173, argues that the distinction of two kinds of plural stems, not from semantic considerations, but from the etymological influence of two distinct Arabic dialects.

For the same reasons, we have not distinguished *Collective* Nouns and non-Collective Nouns. We have not covered *M-Infinitives*, *Nouns of the Verb*, *Comparatives*, and *Exaggerative Nominals*. Other categories that we have not covered are *Nouns of Origin*, *Dimunitives*, certain rare BP-patterns, *Free Accusative Pronouns, Relative Pronouns*, and *Nouns of Instance* and of *Manner*. All these we have not included because of limitations of database size, in relation to time and space. However, we believe that the methods used for the sample CATEGORIES of Nominal—like those used for Verbs—can be generalized to those that were not covered here.

II A FORMAL ACCOUNT OF THE NOMINAL IN M.S.A.

II.1 GENERALIZATION

The automatic processing of Arabic Nominals—either for generation or for recognition—can be approached in different ways. Like processing Verbs, processing Arabic Nominals has to handle data representation and analysis. Here again, we ask: can we adopt a dictionary approach to the representation of the above data on Arabic Nominals, and list all their possible forms and allomorphic variations? Or, should we rather adopt a purely programming approach that relies on processing rules and procedures? Or, should we try to reconcile the two approaches? If we were to adopt the dictionary approach and list all the different NUMBER and GENDER categories for each stem, in each CASE, FLEXION, and DEFINITENESS category, we would obviously end up with a huge and uneconomic table of data. Each Nominal stem has at least two GENDER forms: masculine and feminine; and three regular NUMBER forms: singular,

dual, and plural (with very few exceptions). In addition, it may have one or more forms in the BP category. All of these forms are also distinguished for at least three CASES: nominative, accusative, and oblique. Each CASE ending has at least two variations for full and partial FLEXIONS: full vowel, or single vowel mark; and full, or reduced NGC suffix. This gives us the following formula for each Nominal Form:

(M, F) S, D, P, BP in (full/partial FLEXION) NOM, ACC, OBL.

A simple multiplication operation tells us that an average of: 2 x 4 x 2 x 3 = 48 different forms can be predicted for each stem. This average will grow exponentially with the addition of each new BP form. But taking 48 as an example, and applying it to the 149 N-Stems selected here, we obtain a total of 7,152 forms, which projected on a lexicon of 3,000 entries would mean a NUMBER/GENDER inflection table with 144,000 entries. Such an approach is both inefficient and impossibly unwieldy; for this reason, as well as on grounds of theoretical elegance, we have to seek a generalization of the data, such that an economic and efficient statement of the different forms of Nominal, together with the derivation rules, will allow the generation of all possible Nominal Forms, without the need for listing all of them. In order to achieve such a generalization, we have to carry out four main tasks:

a. Identify the boundaries of the N-Patterns.

b. Identify the stem form within NFs.

c. Identify the relevant N-Affixes.

The tasks of identifying and outlining the distribution of these N-Pattern, stem, and affix forms have to be performed in such a way as to arrive at canonical forms for those structures, in order to achieve economy and cost-effectiveness.

d. Find a simple method for ASSOCIATion between N-Patterns and their appropriate stem, or stems, in the database.

The identification of canonical NUMBER/GENDER patterns involves the SEGMENTation of NFs in such a way as to isolate affix constituents from pattern forms. At the outset, we can distinguish six classes of pattern: 1) Active Verbal, 2) Passive Verbal, 3) Substantive, 4) Tlocative, 5) Numeral, and 6) Adjective. However, not every stem has derivational patterns for all six classes.

For each of these classes, we can distinguish two NUMBER types of pattern for Nominals: 1) a *Base* (initially singular) type, and 2) a *BP* type. Derivation from the Base-pattern type involves the use of a Base pattern with or without NUMBER, GENDER, and CASE suffixes. Derivation from the BP-pattern type involves the use of a BP-pattern with CASE suffixes, but not NUMBER, or GENDER suffixes, the NUMBER value being achieved through vocalic

changes to the Base pattern—rather than affixation—and the GENDER value being determined by that of the source Base pattern.

Within each of the above six classes of pattern, we can also distinguish three different GENDER types of pattern for Nominals: 1) *Masculine-Feminine* (*MF*), or neutral of GENDER, 2) *Masculine Only* (*MO*), and 3) *Feminine Only* (*FO*).

Thus, we can account for suffix variation independently by eliminating suffixes from pattern representations. The isolation of such suffixes allows the collapsing of the various NUMBER/GENDER forms of pattern—for each stem—into a single form for listing in the database. The isolation of Nominal suffixes also allows the differentiation of the contextual conditions needed for specific CASE, FLEXION, NUMBER, and GENDER ASSIGNment.

To summarize the above analysis, we can say that for each stem, we need two different pattern forms: Base and BP forms. The Base pattern can be used for the generation of (masculine, feminine) singular Nominals; and for (Masculine, Feminine) Regular Pluralization and dualization. The BP-pattern is used to derive Broken Plurals. The precise form of the Base- and BP-pattern types depends on the CATEGORY of the Nominal in question.

II.1.1 Identification and Distribution of the Nominal Patterns

In Chapter 1, § II.3.5, we argued for a simplification of the conventional method of pattern representation from the old format 'faɛala-yafoɛalu' to the new format 'cacac-acocac'. Here again, we use this new format for a simpler and uniform statement of the various pattern boundaries and of pattern distribution for the different Nominals. Owing to the lack of space and for economy of exposition, we shall illustrate this distribution in the form of lists (given in App. A, § B).

II.1.1.1 Verbal Patterning

II.1.1.1.1 The Patterning of Active Verbals

List 1 (in App. A, § B) shows the distribution of Active Verbal patterns for each Verbal root, in the Base- and in the BP-pattern types; and in three parts: MF, MO, and FO.

We notice that the forms in List 1 fall into two main groups:

a. A small irregular group including the Basic and Augmented Df.Wk. roots: {bn, t-n, m:, d-k, gn, svq, ns, h-d, :x, bnn, t-nn, gnn, d-kk}. This group serves to derive Df.Wk. Verbals,

and will therefore be restricted as to the type of CASE inflection suffix it can take. This affects the Base patterns 13, 14, 15, and 25; and the BP-pattern 28.

b. A large regular group including all stems except those in *a.* above. This group serves to derive non-Weak (i.e., Abbreviated and Extended) types of Verbal, and will not be restricted as to the type of CASE-inflection suffix it can take.

II.1.1.1.2 The Patterning of Passive Verbals

List 2 (in App. A, § B) shows the distribution of Passive Verbal patterns for each Verbal root, in the Base and in the BP types; and in three parts: MF, MO, and FO.

The patterns given in List 2 are of two different types:

a. Regular patterns including the Base patterns 1 to 8, 11, and 12.

b. A small group of irregular patterns including the Base patterns 9 and 10, and the BP-patterns 11 and 12. This group serves to derive Passive Verbals from the Base and Augmented Defective roots: {:x, bnn, t-nn, gnn, d-kk}. All the patterns in this group end in the Ab.N suffix 'ay'.

II.1.1.2 Non-Verbal Patterning

II.1.1.2.1 The Patterning of Substantives

List 3 (in App. A, § B) gives all the Primitive Substantive patterns for the Nominal stems in the corpus. We notice that not every stem has Substantive derivational patterns. The patterns are listed as usual: in the Base and the BP types; and in three parts: MF, MO, and FO.

In this group of Substantive patterns, we have also included the Derived Substantive pattern 15, which is preceded by the prefix <m> of the Instrumental pattern type. We have argued above for the integration of Primitive Substantives, Derived Substantive Instrumentals, and Tlocatives into the morphological CATEGORY Noun. Another interesting example in the patterns given in List 3, is the Df.Wk. stem 'ns', "neglect", which has the BP-pattern 61, but is the only Nominal stem that has no Base-pattern type.

II.1.1.2.2 The Patterning of Tlocatives

List 4 (in App. A, § B) has a set of 17 Derived Substantive Tlocative patterns, which are distributed for the different Nominal stems that do have Tlocative derivational patterns. Each of these patterns is preceded by the Tlocative prefix <m>. Of interest here, are the Df.Wk.

stems: {bn, t-n, gn}. These have a Base pattern 12, which is of the Abbreviated type, and which has the suffix 'ay' included in it. When converted to the BP type, this pattern becomes a Weak one, which can have the suffix 'iy' attached to it, under certain conditions. These conditions are described in detail in Chapter 7. The patterns given in List 4 are divided as usual into Base and BP types; and into three parts: MF, MO, and FO.

II.1.1.2.3 The Patterning of Adjectives

List 5 (in App. A, § B) has a total of 31 derivational Qualificative Adjective patterns, which are distributed for the different Nominal stems that do have such patterns. Most of these are regular patterns of the 'caciyc' group. However, there are two further groups that deserve notice.

a. A group consisting of the BP-patterns in 13, the second BP-pattern in 14, and the Base pattern 28. All these fall in the Ab.N category.

b. A group consisting of the Base patterns 25, 30, and 31, which fall in the Ex.N category.

Both Ab.Ns and Ex.Ns have different rules of inflection. Of this more below. The 31 patterns in List 5 are divided, as before, into Base and BP-pattern types; and into three parts: MF, MO, and FO.

II.1.1.2.4 The Patterning of Numerals

List 6 (in App. A, § B) gives 15 Derived-Substantive-Numeral patterns, which are distributed for the different Nominal stems that do have such patterns. All the 15 patterns are regular ones except for the Base pattern 7, which is of the Df.Wk. type, and the Base pattern 14, which is of the Df.Ab. type. The patterns given in List 6 are divided, as before, into Base and BP-pattern types; and into three parts: MF, MO, and FO. But in addition to these, there is also the Base pattern 15, which exclusively serves to form dual Numerals.

II.1.2 Formation of NUMBER and GENDER Categories from the Nominal Patterns and Graphotactic Affixation Conditions

Each type of pattern—in all the Nominal CATEGORIES discussed above, and including (Active and Passive) Verbals, (Substantive, Instrumental, and Tlocative) Nouns, Adjectives, and Numerals—can be used either independently, or with the affixation of Nominal suffixes, in order to generate NUMBER categories, such as the singular, the dual, and the plural. Each NUMBER category can be generated for the masculine or the feminine GENDER category. The NUMBER and GENDER suffixes may also carry CASE and FLEXION values. However, the

affixation of the suffixes to Nominals is subject to Graphotactic Conditions, and may possibly depend also on Nominal stem type.

II.1.2.1 Generation of the Masculine Singular Category

The generation of the masculine singular category, for a Nominal in Arabic, involves the use of any Base pattern type in the MF- or the MO-groups, but never the FO-group. Normally, this derivational operation is the realization of the abstract pattern combined with the actual lexical stem (and it is to this combination that CASE-only suffixes—specified below—are affixed). However, this derivational process applies only to So.Ns.

The formation of the masculine singular category from Df.Ab. patterns, such as Base patterns 9 and 10 in List 2, and Base pattern 12 in List 4, does not involve any affixation of CASE suffixes; whereas the application of this process to Df.Wk.Ns, such as Base patterns 13, 14, 15, and 25 in List 1, and Base pattern 7 in List 6, involves the suffixation of the irregular and restricted set of CASE suffixes: {i+, iy}. There are no Df.Ex. patterns in the masculine singular category.

II.1.2.2 Feminization in the Singular

The generation of the feminine singular for Arabic Nominals involves the use of any Base Pattern type in the MF- or the FO-groups, but never the MO-group. This generation is achieved in one of two ways:

1) The use of the derivational pattern alone—without a GENDER suffix—for Nominals that are feminine by convention or by sex; and

2) The use of the derivational pattern and the feminine GENDER suffix 'at' for grammatical or morphological *feminization*.

Both of these types then attach with CASE suffixes. Further, this derivational process applies to So.Ns and Df.Ns alike, with certain exceptions:

a. The feminization of Base patterns 28 (List 5) and 14 (List 6)—which represent Df.Ab.Ns—does not involve the affixation of CASE suffixes.

b. The feminization of Base patterns 11 and 12 (List 2), for Weak stems, involves the addition of the GENDER suffix 'a|t' to the patterns, which then accept CASE suffixes.

c. The feminization of Base pattern 46 (List 3) involves the insertion—in the pattern—of the suffix 'ah', before the GENDER suffix 'at'.

d. The feminization of Base patterns 25, 30, and 31 (List 5)—which represent Df.Ex.Ns—involves the insertion, in the pattern, of the Extended suffix 'a|:', which can then be followed

by CASE suffixes.

e. The feminization of Base patterns 62 and 64 (List 3) involves the use of the GENDER suffix 'ot', which can be followed by CASE suffixes.

f. The feminization of Base pattern 7 (List 3) involves the following simple transformation:

<:> $\rightarrow\!\!\!\!\rightarrow$ <y> / [ ] 'cica|:' [+feminine].

II.1.2.3 Dualization

The *dualization* of any singular masculine or feminized Nominal, in Arabic, can be achieved only through the use of the dual NUMBER/CASE suffixes: {a|, a|ni, ayo, ayoni}. In addition, the dual Base pattern 15 of List 6, can be used only to generate dual Numerals. The affixation of the dual suffixes to So.Ns is *context free*, i.e., subject to no restrictions. However, their affixation to Df.Ns is subject to Graphotactic Conditions, which force simple transformations at the boundaries between pattern and suffix as follows:

a. The dualization of Df.Wk.Ns, such as those represented by Base patterns 13, 14, 15, and 25 (List 1) and Base pattern 7 (List 6), is subject to the insertion of the Df.Wk. suffix 'iy', but not <i+> as follows:

P/S $\rightarrow\!\!\!\!\rightarrow$ P/S$'iy' / [ ] P/S [+dual]; but *P/S$<i+> / [ ] P/S [+dual]; (where P/S is the realization of a pattern/stem combination).

Thus, the pattern 'ca|c' can form the masculine singular Verbal 'ba|ni+', "a builder", the masculine singular-plural Verbal 'ba|niy', "builder(s)", and the masculine dual 'ba|niya|ni', "two builders", but not '*ba|ni+a|ni'. The epenthetic glide <y> is reinserted, after the simple transformation of the full vowel <i+> to a single vowel <i>, in order to avoid hiatus between the <i> and the <a>.

b. The dualization of Df.Ex.Ns, such as those represented by Base patterns 25, 30, and 31 (List 5), is subject to this simple transformation:

<:> $\rightarrow\!\!\!\!\rightarrow$ <w> / [ ] P/S [+dual].

Hence, the pattern 'cacoca|:' can derive the feminine singular Adjective 'zaroqa|:', "blue", and the feminine dual Adjective 'zaroqa|wa|ni', "two blue ...", but not '*zaroqa|:a|ni'.

c. The dualization of Base patterns 34 and 35 (List 3) is subject to the insertion of the suffix 'aw'—thereby restoring the deleted semivowel <w> of the Defective stems {:b, :x}—as follows:

P/S $\rightarrow\!\!\!\!\rightarrow$ P/S$'aw' / [ ] P/S [+dual].

Hence, we can form the masculine singular Substantives ':abu+', "a father", and ':axu+',

"a brother", from the pattern 'cac'. We can form the masculine dual Substantives ':abawa|ni', "two fathers", and ':axawa|ni', "two brothers", but not '*aba|ni', and '*:axa|ni'.

d. Finally, the feminine Base pattern 14 (List 6) is inhibited from the formation of a dual form, since there is no Arabic Numeral '*:ih-odaya|ni' from the Defective stem 'h-d', "one".

II.1.2.4 Masculine Regular Pluralization

The *Masculine Regular Pluralization* of any MF- or MO-Nominal pattern in Arabic, is restricted to Qualificatives of humans, such as Verbals and Adjectives, as well as certain Numerals. In addition, Substantives which are derived using Base patterns denoting humans, are allowed to form *Masculine Regular Plurals* (*MRPs*). Hence, Tlocatives are automatically excluded from this category. Such Masculine Regular Pluralization involves the use of the RP NUMBER/CASE suffixes: {uw, uwna, iy, iyna}, which can also take on the irregular allomorphic variation forms: {awo, awona, ayo, ayona}, in certain contexts, depending on Nominal and stem types. The affixation of MRP affixes to So.Ns is not subject to any restrictions other than the above. However, their affixation to Df.Ns is subject to Graphotactic Conditions and simple boundary transformations as follows:

a. The pluralization of Df.Wk.Ns, such as those represented by Base patterns 13, 14, 15, and 25 (List 1) and Base pattern 7 (List 6), is subject to the deletion of the Df.Wk. suffixes: {i+, iy}, as follows:

P/S $\rightarrow\!\!\!\!\rightarrow$ P/S$0 / [ ] P/S [+MRP].

P/S $\rightarrow\!\!\!\!\rightarrow$ *P/S${i+/iy} / [ ] P/S [+MRP].

Thus, the pattern 'ca|c' can form the MRPs 'ba|nuwna' and 'ba|niyna', "builders", but not '*ba|niyuwna' or '*ba|niyiyna'.

b. The pluralization of Df.Ab.Ns, such as those represented by Base patterns 9 and 10 (List 2) is subject to simple boundary transformations to the Df.Ab. suffix: 'ay', as follows:

'ay' $\rightarrow\!\!\!\!\rightarrow$ 'awo'/'awona' / [ ] P/S [+MRP], in NOM; and

'ay' $\rightarrow\!\!\!\!\rightarrow$ 'ayo'/'ayona' / [ ] P/S [+MRP], in ACC and OBL.

c. The pluralization of pattern 64 (List 3) is an exceptional departure from the norm, in that a feminine Base pattern is used for the formation of the MRP Nominal. This formation is subject to a simple transformation inside the pattern as follows:

'cic' $\rightarrow\!\!\!\!\rightarrow$ 'cac' / [ ] 'cic' [+MRP].

d. The pluralization of Base pattern 8 (List 6) runs along similar lines to case *c.* above, in that there is a simple transformation inside the pattern to form the MRP Numeral:

'cacoc' $\rightarrow\!\!\!\!\rightarrow$ 'cicoc' / [ ] 'cacoc' [+MRP].

e. The pluralization of Numerals involves the use of only the subset: {uwna, iyna} from the plural suffixes, since the Numerals do not occur in Annexed or Modifier positions, which would require the inflection CASES implied by the suffixes {uw, iy}. The other plural allomorphs are not involved since there are no Abbreviated Numerals for pluralization.

II.1.2.5 Feminine Regular Pluralization

The *Feminine Regular Pluralization* of any MF- or FO-Nominal pattern in Arabic is restricted to Qualificatives of humans—such as Verbals and Adjectives—as well as certain Numerals, Substantives, and Tlocatives which are derived from Base patterns, plus certain Infinitives and conventional cases. This process of pluralization involves the use of the *Feminine Regular Plural* (*FRP*) suffix 'a|t'. The affixation of this suffix to So.Ns is free of restriction. However, its affixation to Df.Ns is subject to graphotactic and simple boundary transformations, as follows:

a. The Feminine Regular Pluralization of the Passive Verbal unusually makes use of the BP-pattern type, in order to form the FRP form, in the case of patterns 11 and 12 (List 2). This is because these patterns, in their singular Base form, already use the plural suffix 'a|t', with the meaning of the singular.

b. The FRP Tlocative is exceptionally formed from the masculine Base pattern type, in the case of patterns 8 and 10 (List 4). Usually this type is restricted to masculine derivation.

c. The Feminine Regular Pluralization of Base pattern 64 (List 3) is subject to the following simple transformation:

'cic' $\rightarrow\!\!\!\!\rightarrow$ 'cac' / [ ] 'cic' [+FRP].

d. The Feminine Regular Pluralization of Base pattern 7 (List 3) is subject to the following simple transformation:

'cica|:' $\rightarrow\!\!\!\!\rightarrow$ 'cica|y' / [ ] 'cica|:' [+FRP].

e. The Feminine Regular Pluralization of Base pattern 62 (List 3) is subject to this simple transformation:

'cuc' $\rightarrow\!\!\!\!\rightarrow$ 'cac' / [ ] 'cuc' [+FRP].

f. The Feminine Regular Pluralization of Base pattern 70 (List 3) is subject to this simple transformation:

'wicoc' $\rightarrow\!\!\!\!\rightarrow$ 'wacac' / [ ] 'wicoc' [+FRP].

g. The Feminine Regular Pluralization of Base patterns 25, 30, and 31 (List 5) involves this simple transformation:

<:> $\rightarrow\!\!\!\!\!+$ <w> / [ ] P/S [+FRP].

h. Finally, pattern 14 (List 6) is inhibited from Feminine Regular Pluralization, since there is no Numeral form: '*:ih-odaya|t'.

II.1.2.6 Broken Pluralization

Broken Pluralization is one of the most frequent processes of pluralization in M.S.A., and one which relies on a vast number of patterns exceeding 30 for the *plural of abundance* alone, with an added four for the *plural of paucity*, according to YAEQUWB, 1986: 241. It is so called *Broken Pluralization*, because, according to traditional views of derivation (cf. Ch. 1), it "breaks" or "fragments" the singular Base pattern form by forcing, in it, either vocalic changes only, or vocalic changes with deletion or insertion of new consonants and vowels. We have referred to such added elements as *Supernumerary* ones.

For the patterns given in Lists 1 to 6, Broken Pluralization involves the use of the BP-pattern type (second column in the Lists), which can then be followed by CASE-only suffixes that are specified below. This process can be applied to the MF- and MO-pattern types in order to generate BPs that are either of neutral or of masculine GENDER; and it can be applied to FO-pattern types to generate BPs of feminine GENDER. However, as noted in § II.1.2.5, the BP-pattern types in 11 and 12 (List 2) are exceptionally and exclusively used for the generation of FRPs, but not FBPs. The process of Broken Pluralization applies—as described above—to So.Ns only. However, in the case of Df.Wk.Ns, represented by BP-patterns 28 (List 1) and 12 (List 4), Broken Pluralization involves the affixation of the irregular and restricted set of CASE suffixes: {i+, iy}.

II.1.2.7 Summary of the NUMBER/GENDER Values of the Nominal Pattern Types

To summarize the above analysis for Nominal-pattern types, we can say that we have distinguished two main derivational-pattern types, and three GENDER groups, with NUMBER values as follows:

a. A Base pattern type consisting of:

i. A MF-group which derives: (M, F) S, D, RP.

ii. A MO-group which derives: (M) S, D, RP.

iii. A FO-group which derives: (F) S, D, RP.

b. A BP-pattern type consisting of:

i. A MF-group which derives: (M, MF) BP.

ii. A MO-group which derives: (M) BP.

iii. A FO-group which derives: (F) BP.

In addition, we have distinguished an exclusively dual pattern, for Numerals only, with the values: (M, F) D. There were some exceptions, in certain cases, to these generalizations as follows:

c. In Base pattern types:

i. A MO-subset which derives: (F) RP.

ii. A FO-subset which derives: (M) RP.

d. A BP-pattern type which derives: (F) RP.

We have also distinguished two kinds of Nominal derivation:

e. The regular derivation of So.N types, which is free of restriction.

f. The irregular derivation of Df.Ns, which is subject to graphotactic and boundary restrictions, as follows:

i. The derivation of Df.Ab.Ns in the singular accepts no CASE suffixes; and in the plural is subject to simple boundary transformations, which cause allomorphic variation in the plural suffix.

ii. The derivation of feminine Df.Ex.Ns in the dual and plural is subject to boundary conditions that cause transformations in the pattern boundaries.

iii. The derivation of Df.Wk.Ns, in the singular and BP, is restricted to the use of the irregular set of CASE suffixes: {i+, iy}. The suffix <i+> is deleted under dualization, and pluralization; whereas the suffix 'iy' is deleted under pluralization only.

In addition, the process of Regular Pluralization is restricted—in the main—to Qualificatives of humans, certain Numerals, and certain exceptional cases. A number of patterns undergo simple transformations under dualization or pluralization. Finally, the pattern ':icocay', for the Numeral, cannot be dualized or pluralized; and the stem 'ns', "neglect", has no Substantive Base pattern.

II.1.3 Inflection of the Nominal for CASE

Besides being inflected for NUMBER and GENDER, Nominals in M.S.A. can also be inflected for CASE. This CASE inflection is achieved, as hinted above, through the use of two types of suffix: purely Vocalic suffixes—which carry CASE-only values—and other suffixes that carry both NUMBER and CASE values at the same time. Otherwise, CASE inflection is neutralized without the use of suffixes.

II.1.3.1 CASE Suffixes

Purely Vocalic suffixes are of two kinds: either single vowel marks or full vowel marks. In general, these vowels have three different kinds of CASE distribution:

a. Unique CASE values, with singular So.Ns and certain BPs.

b. Syncretic CASE values, where an accusative CASE MARK fills in for the oblique CASE, or vice versa, with certain Diptote Nominals.

c. Exceptional suffix forms used with Df.Wk.Ns.

Table 39 below lists all the Vocalic CASE suffixes of Arabic Nominals, and gives their exact CASE values and distribution.

DECLINA-BILITY	CASE	SINGLE VOWEL	FULL VOWEL	NOMINAL TYPE/PATTERN
Fully Declinable of Type 1	*NOMINATIVE* *ACCUSATIVE* *OBLIQUE*	u a i	u+ a+ i+	singular So.Ns, BP-patterns that are not in the third row below.
Diptote of Type 2	*NOMINATIVE* *ACC-OBL*	u i	u+ i+	FRPs.
Diptote of Type 1	*NOMINATIVE* *ACC-OBL*	u a	u+ a+	PNs ending in 'at', or having the pattern: cawa|cic; all the Hyper BP-patterns: maca|cic, :ama|cic maca|yic, mawa|cic, cawa|cc, :aca|cic, cawa|cic, cawa|:ic, caca|ciyc, maca|ciyc; & the singular Adjective patterns: :acocac, :acowac, :acoyac.
Diptote of Type 3	*NOM-OBL* *ACCUSATIVE*	iy iya	i+ iya+	Df.Wk.Ns ending in 'iy'.

TABLE 39: Vocalic CASE Suffixes of the Nominal in M.S.A.

II.1.3.2 CASE-AND-NUMBER Suffixes

CASE-AND-NUMBER suffixes are of two kinds, either full or reduced. In general, these suffixes are reserved for the dual, and MRP; and have a two-way distribution depending on CASE:

a. A unique suffix for the nominative; and

b. A syncretic suffix shared between the accusative and the oblique.

However, these suffixes undergo allomorphic variation with certain types of Df.Ns. We have added to this group of Nominals, those Nominals that do not take any CASE suffixes, and indicate by themselves NUMBER and CASE values. These include Ab.Ns and certain Pronouns.

Table 40 below lists all the CASE-AND-NUMBER suffixes of Arabic Nominals, and gives their exact values for CASE and for NUMBER, as well as their distribution. Note that Table 40 includes Nominals that are fully declinable of Type 2, whereas those that are of Type 1 were included in Table 39.

DECLINABILITY	CASE	SUFFIX		NOMINAL TYPE
		REDUCED	FULL	
Fully Declinable of Type 2				So.Ns:
	NOMINATIVE	a|	a|ni	dual.
		uw	uwna	plural.
	ACC-OBL	ayo	ayoni	dual.
		iy	iyna	plural.
				Df.Wk.Ns:
	NOMINATIVE	iya|	iya|ni	dual.
		uw	uwna	plural.
	ACC-OBL	iyayo	iyayoni	dual.
		iy	iyna	plural.
				Df.Ab.Ns:
	NOMINATIVE	aya|	aya|ni	dual.
		awo	awona	plural.
	ACC-OBL	ayayo	ayayona	dual.
		ayo	ayona	plural.
Indeclinable	*NOMINATIVE*	a|	a|ni	dual DPs.
		Ø	Ø	SPs.
	ACC-OBL	ayo	ayona	dual DPs.
	OBLIQUE	Ø	Ø	DPs.
	NEUTRAL	Ø	Ø	Df.Ab.Ns ending in 'ay'; PNs ending in 'ay'; singular & plural DPs.

TABLE 40: CASE-AND-NUMBER Suffixes of the Nominal in M.S.A.

II.1.4 Summary of the NUMBER, GENDER, CASE, and FLEXION Values of the Nominal Suffixes

We can deduce from the analysis in § II.1 above, that a *Nominal Form* (*NF*) is the realization of a shallow—or semi-abstract—pattern, that is filled with a discontinuous lexical root or stem; and to which a number of suffixes can be affixed, as follows:

a. A GENDER suffix (G) for the morphologically feminized category only.

b. A CASE suffix (K)—following the GENDER suffix, if any—for the singular and the BP categories, and for FRPs.

c. A CASE-AND-NUMBER suffix (K)—after the GENDER suffix, if any—in the dual category.

d. A CASE-AND-NUMBER suffix (K) in the MRP category.

In addition, we can deduce from § II.1.3, a general division of the NGC suffixes into three categories: singular, dual, and plural:

i. G = {at, ot} [+sn]; and K = {u, u+, a, a+, i, i+} [+sn].

ii. K = {a|ni, a|, ayoni, ayo} [+dl].

iii. G = {a|t} [+pl]; and K = {uwna, awona, iyna, ayona, uw, awo, iy, ayo} [+pl].

The above statements can be expressed in the form of a constraint that restricts the juxtaposition of G to K as follows:

*G$K, where G [+sn] and K [+pl]; or G [+pl] and K $\neq$ [+sn]/K [+acm].

Note that the above constraint inhibits *G$K for a plural G and an accusative K, because of the declinability restrictions on FRPs.

The affixation of CASE suffixes and of CASE-AND-NUMBER suffixes, to Df.Ns, is subject to graphotactic restrictions, and/or allomorphic variation. Besides carrying NUMBER, GENDER, and CASE values, the suffixes have FLEXION types, which are of relevance to the definitization process of Nominals in M.S.A. The NUMBER, GENDER, CASE, and FLEXION values are summarized in Table 41 below.

II.1.5 Identification and Distribution of the Nominal Roots and Stems

Having identified Nominal patterns and Nominal NGC suffixes, we can now turn our attention to the identification of the third constituent of a Nominal Form, which we specified—in § II.1.4 above—as being the discontinuous root or stem, which occurs at the edges of syllables, within the pattern realization. In § I.1.3.1 above, we argued for a formal distinction between Verbal

SUFFIX FORM		FLEXION	NGC VALUES
DEFAULT	CONTEXTUAL		
NGC SUFFIXES			
u		svm	(M, F) S, BP & (F) RP in NOM.
u+		fvm	(M, F) S, BP & (F) RP in NOM.
a		svm	(M, F) S, BP in ACC & OBL.
a+		fvm	(M, F) S, BP in ACC.
i		svm	(M, F) S, BP in OBL & (F) RP in ACC & OBL.
i+		fvm	(M, F) S, BP in NOM & OBL & (F) RP in ACC & OBL.
iy		svm	(M, F) S, BP in NOM & OBL.
iy	ayo	dnf	(M) RP in ACC & OBL.
iyna	ayona	nnf	(M) RP in ACC & OBL.
uw	awo	dnf	(M) RP in NOM.
uwna	awona	nnf	(M) RP in NOM.
a|		dnf	(M, F) D in NOM.
a|ni		nnf	(M, F) D in NOM.
ayo		dnf	(M, F) D in ACC & OBL.
ayoni		nnf	(M, F) D in ACC & OBL.
GENDER SUFFIXES			
at			(M, F) BP &
	ot, a|t	N/A.	(F) S.
a|t		N/A.	(F) RP & (M) BP.
a|:		N/A.	(M, F) S, BP.
ay		xvm.	(M, F) S & (M) BP in NEUT.

TABLE 41: NUMBER, GENDER, CASE, and FLEXION Values of the Nominal Suffixes

roots, and Nominal stems. However, we need to look more closely at Nominal radical structure in order to be able to define a canonical form for each root, rather than simply list all possible radical forms.

II.1.5.1 Verbal Roots

Verbal roots are different from Nominal stems and more similar to Verb roots in three respects:

a. They carry TRANSITIVITY values; and

b. Their TRANSITIVITY value can be modified through causativization and passivization. This means that Augmented roots are divided into intensive roots—which keep the same TRANSITIVITY—and causative ones, which are PROMOTed to higher TRANSITIVITY ranks. Passivization DEMOTes roots to lower TRANSITIVITY ranks.

c. They are semantically Derivative—in that they are used in the semantic sense of the root that is normally used for a Verb—i.e., action, state, process, or event.

Verbal roots are much more variable than Verb roots. If we factor out the suffixes and vowels which are in the pattern realization, we are left with raw root forms. The variation, in the raw roots, ranges from initial consonantal forms, to Augmented forms, through intermediate forms that are characteristic of Base- and BP-pattern realizations. In each of the forms, the changes can occur in one of several ways:

- The insertion of the initial Verbal prefix <m> or the Verbal infix <l>.
- The addition of a number of Supernumerary characters from among the set: {s, :, l, t, m, n, y, h, l}.
- The deletion or modification of initial, medial, or final semivocalic radicals—in certain Defective roots—such as the modification of medial radicals, from <w> or <y> to <:>.

Tables 42 and 43 give the details of root form variation for Defective and Sound Verbals, respectively.

II.1.5.2 Non-Verbal Stems

Non-Verbal stems—in which we include the stems of Substantives, Tlocatives, Adjectives, and Numerals—are different from Verb and Verbal roots in two respects:

a. They carry no TRANSITIVITY values, and therefore have no Augmented forms with modified semantics. In other words, they may have doubled consonants, but this is not a regular process of augmentation with regularized or predictable consequences.

b. They are semantically Primitive. In particular, a stem is used in a semantic sense, which is normally divorced from its use as a root in a Verb or in a Verbal, i.e., it denotes an entity not an action, event, etc.

Non-Verbal stems are much more variable than Verb and Verbal roots, since they have a large number of BP-patterns which involve the insertion of one or more characters from the Supernumerary character set: {s, :, l, t, m, n, y, h, l}. In particular, the variation in stem form occurs in one of several ways:

INITIAL ROOT FORM	ALTERNATIVE ROOT FORMS		
	IN BASE PATTERNS	IN AUGMENTED PATTERNS	IN BROKEN PLURAL PATTERNS
ld	w|ld, mwlwd	mwlld	
rd	w|rd, mwrwd	mwrrd	
h-d	h-|d, mwh-wd, h-|dy	mwh-h-d	
sm	w|sm, mwswm	mwssm	
:l	:|wl	m:wwl	
kn	k|:n, mkwn	mkwwn	kw|:n
qm	q|:m, mqwm	mqwwm	qyy|m, qyym, qw|:m
t;l	t;|:l	mt;wwl	
bb		mbwwb	
jd	j|:d	mjwwd	
xl	x|:l	mxwwl	
s:		msww:	
sd	s|:d, mswd	mswwd	s|dt
s;r	s;|:r	ms;yyr	
bt	b|:t	mbyyt	
bd;	b|:d;	mbyyd;	
bn	b|n, b|ny, mbnyy,	mbnn, mbnny, mbnn|t	bn|t, bw|n
t-n	t-|n, t-|ny, mt-nyy	mt-nn, mt-nny, mt-nn|t	t-w|n
m:	m|:, m|:y, mm:yy		
d-k	d-|k, d-|ky	md-kk, md-kky, md-kk|t	
gn	g|n, g|ny	mgnn, mgnny	gw|n, mgnn|t
svq	sv|q, sv|qy		svw|q
:x	m:|x, m:|xy, m:|x|t		m:|xy
ns	n|s, n|sy, mnsyy		nw|s

TABLE 42: Defective Root Forms for Verbals

- The prefixation of one character from the set: {:, m};
- The suffixation of one character from the set: {:, t, n, h};
- The infixation, at the second position from the beginning or from the end of the raw stem form, of the character <:>;
- The prefixation of <:> and the suffixation of one character from the set: {:, t} at the same time; and
- The prefixation of ':m' and the suffixation of <t> at the same time.

INITIAL ROOT FORM	ALTERNATIVE ROOT FORMS		
	IN BASE PATTERNS	IN AUGMENTED PATTERNS	IN BROKEN PLURAL PATTERNS
d/nn	d/|nn, md/nwn		
jdd	j|dd	mjddd	
ɛmm	ɛ|mm	mɛmmm	ɛw|mm
:mm	:|mm	m:mmm	
h-sb	h-|sb, mh-swb		h-sbt
h-sn	h-sn	mh-ssn	
krm	k|rm	mkrrm	
kbr	k|br	mkbbr	
kt-r	k|t-r	mkt-t-r	
s;gr	s;|gr	ms;ggr	
qs;r	q|s;r	mqs;s;r	qw|s;r
jml	j|ml	mjmml	
fqr		mfqqr	
t-mn	t-|mn, mt-mwn	mt-mmn	t-w|mn
svrf	sv|rf	msvrrf	svrf, svrwf, svw|rf
ktb	k|tb, mktwb	mkttb	ktt|b, ktbt
svkr	sv|kr, msvkwr		svkkr
drs	d|rs, mdrws	mdrrs	
frsv	f|rsv, mfrwsv	mfrrsv	
qtl	q|tl, mqtwl	mqttl	qtt|l, qtlt
rbɛ	r|bɛ, mrbwɛ	mrbbɛ	rw|bɛ
skn	s|kn, mskwn	mskkn	skk|n, sw|kn
s;dq	s;|dq	ms;ddq	
nfd-	n|fd-	mnffd-	nw|fd-
jbl	jbl, mjbwl	mjbbl	
h-mr	h-|mr	mh-mmr	
zrq	z|rq		
fth-	f|th-, mftwh-	mftth-	fw|th-
mnh-	m|nh-, mmnwh-		
mnɛ	m|nɛ, mmnwɛ		mnɛt, mw|nɛ
d-hb	d-|hb, md-hwb		d-w|hb
xd;r	x|d;r	mxd;d;r	
jls	j|ls, mjlws	mjlls	jlws

TABLE 43: Sound Root Forms for Verbals

ksr	k|sr, mkswr	mkssr	kssr, kw|sr
d;rb	d;|rb, md;rwb	md;rrb	d;w|rb
nzl	n|zl, mnzwl	mnzzl	nw|zl
h-ml	h-|ml, mh-mwl	mh-mml	h-mlt, h-w|ml
qlm	q|lm, mqlwm	mqllm	qlmt
svbk	sv|bk, msvbwk	msvbbk	
ɛsvr	ɛ|svr, mɛsvwr	mɛsvsvr	ɛw|svr
sds	s|ds, msdws	msdds	
sbɛ	s|bɛ, msbwɛ	msbbɛ	sbɛt, sw|bɛ
t-lt-	t-|lt-, mt-lwt-	mt-llt-	t-w|lt-
tsɛ	t|sɛ, mtswɛ		
xms	x|ms, mxmws	mxmms	xw|ms
:lf	:|lf, m:lwf	m:llf	:ll|f, :w|lf
s;fr	s;|fr	ms;ffr	
h-dq	h-|dq, mh-dwq	mh-ddq	
fhm	f|hm, mfhwm	mfhhm	
ɛlm	ɛ|lm, mɛlwm	mɛllm	ɛll|m
qdm	q|dm, mqdwm		qdm, qdwm, qw|dm
svrb	sv|rb, msvrwb	msvrrb	svrb, svw|rb
rkb	r|kb, mrkwb	mrkkb	rkk|b, rkb|n, rw|kb
mr:	m|r:, mmrw:	mmrr:	
rjl	r|jl	mrjjl	rjj|l, rjl|n

TABLE 43: Sound Root Forms for Verbals (Contd)

In addition, Defective stems undergo deletion or modification of their radical semivowels. The simplification of stem-form variation, in order to obtain a canonical stem form for database storage, can be achieved through:

i. The elimination of the vowels in the pattern realization and of suffixes of prolongation such as 'a|:'; and

ii. The elimination of all semivowels since they are irrelevant: there are no TRANSITIVITY values, even for stems with doubled or duplicated semivowels.

iii. The elimination of the Supernumerary characters at the positions predicted immediately above. However, the problem is when those characters coincide with a character that is a constituent radical of the stem. This particular problem is solved at the level of computational processing.

In Tables 44 and 45 below, we give the exact details of radical variation for each non-Verbal

stem, first for Defective stems and then for Sound stems.

INITIAL STEM FORM	ALTERNATIVE STEM FORMS	
	IN BASE PATTERNS	IN BROKEN PLURAL PATTERNS
ld	wld, wlyd, wl|d, mwld	:wl|d, wldt, wld|n, wl|:d, mw|ld
rd	wryd, wrd, mwrd	:wrdt, wrwd, :wr|d, mw|rd
h-d	wh-d, wh-yd, w|h-d, :h-d, :h-dy	:h-|d
sm	ws|m, wsym, mwsm	:wsmt, wsm|:, mw|sm
:l	:wwl, :wly	:w|:l, :wl
kn	kwn, mk|n	:kw|n, :m|kn, :mknt
qm	qwm, qm, qwym, mq|m	:qw|m, qym, qy|m
t;l	t;wyl	t;w|l, t;y|l
bb	b|b	:bw|b, :bwbt
jd	jw|d, jyyd	:jy|d, jy|d, jy|:d
xl	x|l	:xw|l, :xwlt
s:	syy:	
sd	syyd, :swd, swd|:	:sy|d, s|dt, swd
s;r	ms;yr	ms;|yr
bt	byt, mbyt	:by|t, bywt
bd;	byd;, :byd;, byd;|:	bywd;, byd;
bn	bn, :bn, bn|:, mbny	:bn|:, :bnyt, mb|n
t-n	:t-n, mt-ny, mt-|n	mt-|n
m:	m:	
d-k	d-kyy	:d-ky|:
gn	gnyy, mgn	:gny|:, mg|n
svq	svqyy	:sv|qy|:
:x	:x, :xw	:x|:, :xwt
:b	:b, :bw	:b|:
ns		ns|:, nswt

TABLE 44: Defective Stem Forms for Non-Verbals

INITIAL STEM FORM	ALTERNATIVE STEM FORMS	
	IN BASE PATTERNS	IN BROKEN PLURAL PATTERNS
d/nn	d/nyn, md/nn	:d/nn|:, md/|nn
jdd	jdd, jdyd	:jd|d, jdwdt, jdd
ɛmm	ɛmm, ɛm|m	:ɛm|m, ɛmwmt, ɛm|:m
:mm	:m|m, :mm, :mmh, :mym	::mmt, :ymmt, :mm, :m|:m
h-sb	h-s|b, h-sb, h-syb, mh-sb	h-sb|n, :h-s|b, h-sb|:, mh-|sb
h-sn	mh-sn	h-s|n, mh-|sn
krm	krym	krm|:, kr|m, kr|:m
kbr	kbyr	kbr|:, kb|r, kb|:r
kt-r	kt-yr	
s;gr	s;gyr	s;g|r, s;gr|:, s;g|:r
qs;r	qs;r, qs;yr, mqs;r	qs;wr, :qs;|r, qs;r, qs;|:r, qs;r|:, qs;|r, mq|s;r
jml	jml, jmyl	:jm|l, jm|l̩jml|:, jm|:l
fqr	fqyr	fqr|:, fq|:r
t-mn	t-mn, t-myn, t-m|ny, t-m|n, mt-mn	:t-m|n, :t-mnt, t-mn, mt-|mn
svrf	svrf, svryf, msvrf	svrf, svrf|:, :svr|f, svr|:f, msv|rf
ktb	ktyb, kt|b, mktb	kt|:b, ktb, mk|tb
svkr	svkyr	svkr
drs	drys, drs, dr|s, mdrs	:dr|s, drs|n, drws, md|rs
frsv	fr|sv, frysv, mfrsv	frsv, fr|:sv, mf|rsv
qtl	qtyl, mqtl	mq|tl
rbɛ	rbyɛ, :rbɛ, mrbɛ	rbɛ, rb|ɛ, rb|:ɛ, mr|bɛ
skn	skyn, mskn	sk|:n, ms|kn
s;dq	s;dyq, s;d|q	:s;dq|:, s;dq|:
nfd-	nfyd-, mnfd-	nfd-, mn|fd-
jbl	jbl, jbyl, mjbl	:jb|l, jb|l, jbl, jb|:l, mj|bl
h-mr	h-myr, h-m|r, :h-mr, h-mr|:, mh-mr	h-m|:r, h-mr, :h-mrt, mh-|mr
zrq	:zrq, zrq|:	zrq
fth-	mft|h-, mfth-	mf|tyh, mf|th-
mnh-	mnyh-, mnh-	mn|:h-, mnh-
mnɛ	mnyɛ	mn|:ɛ
d-hb	d-hb, d-hyb, md-hb	:d-h|b, d-h|b, d-h|:b, md-|hb

TABLE 45: Sound Stem Forms for Non-Verbals

xd;r	:xd;r, xd;r|:, mxd;r	xd;r, mx|d;r
jls	jlys, mjls	jls|:, jll|s, mj|ls
ksr	ksyr, mksr	ksry, ks|ry, mk|sr
d;rb	d;ryb, md;rb	d;rb|:, d;r|:b, md;|rb
nzl	nzyl, mnzl	nzl|:, mn|zl
h-ml	h-myl, mh-ml	h-m|:l, mh-|ml
qlm	qlm, mqlm	:ql|m, ql|m, mq|lm
svbk	svbb|k, svbk	svb|byk, svb|k
ɛsvr	ɛsvyr, ɛsvr, mɛsvr	:ɛsvr|:, ɛsvr|:, ɛsv|:r, mɛ|svr
sds	sdys	:sd|s
sbɛ	sbɛ, sbyɛ, msbɛ	sbwɛ, sb|ɛ, :sb|ɛ, ms|bɛ
t-lt-	t-lyt-, t-l|t-, mt-lt-	:t-l|t-, mt-|lt-
tsɛ	tsyɛ, tsɛ	tsɛ
xms	xmys, xms	:xms|:, :xmst
:lf	:lyf, :lf	:lf|:, :l|:f, :l|f, :lwf
s;fr	s;fyr, :s;fr, s;fr|:	s;fr
h-dq	h-dyq, h-dq	h-d|:q, :h-d|q, h-dq
ɛlm	ɛlm, ɛlym, mɛlm	:ɛl|m, ɛl|m, ɛlm|:, mɛ|lm
qdm	qdm, qdym, mqdm	:qd|m, qd|m, qd|:m, qdm|:, qd|my, mq|dm
svrb	svr|b, svryb, msvrb	:svrbt, msv|rb, svrb|:, svr|b
rkb	rk|b, rkyb, mrkb	rk|:b, rkb, mr|kb
mr:	mr:, :mr:, mry:	mrw:, :mr:t
sɛd	s|ɛd, sɛyd	sw|ɛd, sɛd|:
stt	stt	
rjl	rjl, rjyl, mrjl	:r|jl, rj|l, rjly, rj|ly, mr|jl

TABLE 45: Sound Stem Forms for Non-Verbals (Contd)

We have listed above all non-Verbal stem forms together, as if there were no differences among them. However, Numeral stems are different from other non-Verbal stems, in that they carry NUMERICAL VALUES which are of immediate relevance in determining the syntactic rules of agreement between Numerals, as Modifiers, and other Nominals. The NUMERICAL VALUES vary depending on the NUMBER of the Numeral in question. The details of this variation are listed in Table 46 below.

II.1.6 Definitization

After obligatory inflection for NUMBER, GENDER, and CASE, Nominal Forms in M.S.A. may optionally be *definitized*. *Definitization* is achieved in one of two ways: prefixation of a

STEM	NUMERICAL VALUE		
	in (M, F) S	in (M, F) D	in (M, F) RP
h-d	1	N/A.	N/A.
t-n	N/A.	2	N/A.
t-lt-	3	N/A.	3 TIMES 10
rbɛ	4	N/A.	4 TIMES 10
xms	5	N/A.	5 TIMES 10
stt	6	N/A.	6 TIMES 10
sbɛ	7	N/A.	7 TIMES 10
t-mn	8	N/A.	8 TIMES 10
tsɛ	9	N/A.	9 TIMES 10
ɛsvr	10	N/A.	10 TIMES 2
m:	100	100 TIMES 2	N/A.
:lf	1000	1000 TIMES 2	N/A.

TABLE 46: NUMERICAL VALUES for Nominal Stems

Determiner, or suffixation of a Genitive Pronoun. The two are mutually exclusive.

II.1.6.1 Affixation of the Determiner ':alo' to Simple NFs

The prefixation of the Determiner to a Simple NF is subject to three types of conditions which are concerned with the DEFINITENESS, and FLEXION of the Nominal, as well as the type of initial character it has:

a. The NF must not be definite already. Definite NFs include exophorically definite NFs—such as SPs, DPs, and PNs—and definitized NFs, such as NFs with affixed GPs. Hence:

D$NF where NF [+def], e.g., ':alokita|buhu', "*the book-his", is illegal.

b. The NF must not have the FLEXION of explicit indefiniteness, which is the full vowel mark. Hence: *D$NF where NF [+fvm], e.g., '*:alokita|bu+', "*the a book", is illegal.

NFs with reduced NGC suffixes do not ordinarily attach with the Determiner. There are two exceptions: Numerals in the dual, such as: ':alo:alofa|', "the 2000", and ':alomi:atayo', "the 200"; and Df.Ns carrying the suffix 'iy'—which is also the reduced form of the RP suffix 'iyna'—such as ':aloba|niy', "the builder". Thus:

*D$NF where NF [+dnf] and NF $\neq$ MM, or NF $\neq$ Df.

c. Arabic characters are divided into two types: so called *Lunar* and *Solar* characters. The reader will remember, from the list of transcriptions given at the beginning of this thesis, our definition of these characters as follows:

- Lunar: {:, b, j, h-, x, ɛ, g, f, q, k, m, h, w, y}.
- Solar: {t, t-, d, d-, r, z, s, sv, s;, d;, t;, d/, l, n}.

Thus, there are graphotactic rules of prefixation which we will consider as part of ASRs. These rules force allomorphic variation in the Determiner ':alo', "the"—depending on which type of initial character the NF has—as follows:

i. D ⟶ ':alo' / — [ ] <1 (NF) = Lunar; and

ii. D ⟶ ':al*C* and *C* = <1 (NF) / — [ ] <1 (NF) = Solar.

Rule *ii.* is motivated by assimilation of *C* to the initial character of NF.

II.1.6.2 Affixation of Genitive Pronouns to Simple NFs

Simple NFs can be attached with Genitive Pronouns. We noted, in Chapter 4, § II.4.1, that most enclitic Pronouns in Arabic are in fact homonymic between accusative and genitive CASE; and that they are Accusative when attached to CFs, and Genitive when attached to NFs, except for the first-PERSON-singular Pronoun, which has a different form for each CASE. Other third-PERSON-dual and plural-TPs were noted to share their form with SPs. We then gave a list, in Table 21, of these enclitic Pronouns and described their overlap, in Table 22.

The affixation of GPs to Simple NFs is subject to four different types of conditions, which are concerned with TRANSITIVITY, DEFINITENESS, FLEXION, and graphemic structure.

a. The affixation of GPs to Verbal NFs is subject to TRANSITIVITY conditions, in a parallel way to the affixation of CPs to Verbs:

i. Intransitive Verbals are not allowed to attach with GPs. Hence:

*NF$GP where NF = L1/L2 and L1/L2 [+ntr], e.g., '*ja|lisuhu', "*sitting-him".

ii. Transitive Verbals can attach with GPs, as in: 'da|risu$hu', "studying-it".

b. The affixation of GPs to non-Verbal NFs is free of TRANSITIVITY restrictions. However, the affixation of GPs to all NFs is subject to the same DEFINITENESS restrictions as in II.1.6.1.*a.*, above.

c. The affixation of GPs to all NFs is subject to FLEXION conditions, in that GPs cannot be affixed to NFs with full vowel marks nor to NFs with full NGC suffixes. Hence:

*NF$GP where NF [+fvm]/NF [+nnf], e.g., '*kita|bu+$hu', "*a book-his", and '*kita|ba|ni$hi', "*two books-his", are illegal.

d. There are other Graphotactic Conditions on the affixation of GPs to Simple NFs, which are concerned with allomorphic GP selection and simple boundary transformations. The se-

lection of GP variants and the boundary transformations—which depend on the graphemic environment—are applied as follows:

∀ NF & GP / [ ] NF$GP,

i. ∀ >2 & >1 (NF) = 'uw' $\Longrightarrow$ 'uw' $\rightarrow\!\!\!\!\rightarrow$ 'iy' & GP = 'ya', where NF is a MRP in NOM and with reduced NGC suffix. Hence, '*mudarrisuw$ya', but 'mudarrisiy$ya', "my teachers". However, 'mudarrisuw$hu', "his teachers", is legal. This constraint has not previously been noted by Arab grammarians.

ii. ∀ >2 & >1 (NF) = 'yo' $\Longrightarrow$ <o> $\rightarrow\!\!\!\!\rightarrow$ Ø, & GP = 'ya', where NF is a dual Nominal in ACC-OBL, and with reduced NGC suffix. Hence, '*ma|nih-ayo$ya'; but 'ma|nih-ay$ya', "my two benefactors". However, 'ma|nih-ayo$hi', "his two benefactors", is legal.

iii. ∀ >2 & >1 (NF) = 'iy' $\Longrightarrow$ GP $\neq$ 'iy', where NF is a Df.Wk.N or a MRP in ACC-OBL and with reduced NGC suffix. Thus, '*ba|niy$iy'; but 'ba|niy$ya', "my builder". However, 'ba|niy$hi', "his builder", is legal.

iv. ∀ NF [+sn]/NF [+BP]/NF [+FRP], & GP = 'iy' $\Longrightarrow$
CASE suffix, K (NF) $\rightarrow\!\!\!\!\rightarrow$ Ø. Hence, '*kutubu$iy'; but 'kutub$iy, "my books", is legal.

v. ∀ >2 & >1 (NF) = 'ay' $\Longrightarrow$ <y> $\rightarrow\!\!\!\!\rightarrow$ <|>, & GP $\neq$ 'iy', where NF is a Df.Ab.N. Thus, '*mabonay$ya', '*mabonay$iy', as opposed to 'mabona|$ya', "my building". Also, '*mabonay$hu', as opposed to 'mabona|$hu', "his building", etc.

vi. ∀ stem (NF) = {:b/:x} & GP $\neq$ 1 (M, F) S $\Longrightarrow$

- NF $\longrightarrow$ NF$'uw'$GP, where NF is in NOM; and
- NF $\longrightarrow$ NF$'a|'$GP, where NF is in ACC; and
- NF $\longrightarrow$ NF$'iy'$GP, where NF is in OBL.

Hence, '*:abu$hu', '*:aba$hu', and '*:abi$hi'; but ':ab$uw$hu', ':ab$a|$hu', and ':ab$iy$hi', "his father".

vii. ∀ NF & GP / [ ] NF$GP, & GP $\neq$ (F) S,

- *if* >1 (NF) = {i/y}, or
- *if* >1 (NF) = <o> & >2 (NF) = <y> $\Longrightarrow$ <2 (GP) = <i>.
- *if* >1 (NF) = <o> & >2 (NF) $\neq$ <y>, or
- *if* >1 (NF) = {a/u/w/|} $\Longrightarrow$ <2 (GP) = <u>.

Hence, '*kita|bi$hu', but 'kita|bi$hi', "his book", is legal.

Rules *i.* to *vii.* are rules of GP selection—which regulate the juxtaposition of vowels and semivowels to other vowels, enforcing simple transformations where applicable—and should be read as follows: "forall GPs affixed to NFs, if the NF ends in 'uw' and has the property values specified in *i.* or if the NF ends in 'yo' and has the property values specified in *ii.*, then the GP allomorph selected must be: 'ya', provided 'uw' is substituted with 'iy' in *i.* and <o> is deleted in *ii.*

If the NF ends in 'iy' and has the property values in *iii.* or if the NF ends in 'ay' and has the property values in *v.*, then the GP allomorph selected must not be: 'iy', provided <y> is substituted with <|>, in *v.* If the NUMBER of the NF is singular, BP, or FRP, and the clitic Pronoun is the first-PERSON-singular GP = 'iy', then the CASE suffix K of NF must be deleted, in *iv.*

In *vi.*, if the stem of the NF is either ':b', "father", or ':x', "brother", and the GP is not in the first PERSON singular, then one of three different Supernumerary suffixes must be inserted before the GP: 'uw'—in the nominative CASE—'a|'—in the accusative CASE—and 'iy', in the oblique CASE.

In all other cases, the GP selected must be different from 'ya' and must observe the formal compatibility rules described in *vii.*; that is, if it is not the feminine singular GP—which is in free affixation to NFs—regardless of their final boundary. This is because, this GP has only one possible variant form as opposed to other GPs which, as listed in Table 21, may have two allomorphs. The selection of a given allomorph is not free, but subject to the said compatibility rules—which are parallel to the compatibility rules stated in II.4.3.*c.* that govern the affixation of CPs to CFs—and which are motivated by pronunciation rules.".

II.1.7 Genitivization: Affixation of Bound Prepositions to Simple and Complex NFs

We have seen, in § II.1.4 above, that a Simple NF is a composite of a discontinuous root or stem and a pattern realization where that root or stem is embedded. This composite is in turn combined with an optional GENDER suffix and an obligatory CASE or NGC suffix. In § II.1.6, we stated that a Simple NF itself can become a Complex NF when an optional definitizer is affixed to it. This definitizer can be a prefixed Determiner—in which case we will call the Complex NF, a *Definite NF* (*DNF*)—or it can be a suffixed GP, in which case we will call the Complex NF, an *Annexed NF* (*ANF*). Now, there is a subset of Arabic Prepositions that we will term *Bound Prepositions*, and which are: 'li', "to, for", 'bi', "at, in, with, for", and 'ka', "as". These Bound Prepositions can be affixed to such Complex NFs as DNFs and ANFs as well as to Simple NFs. This process of affixation—which we refer to as *GENITIVIZATION*—is

subject to two different conditions:

a. The following Graphotactic Condition affects the juxtaposition of Bound Prepositions to Determiners as follows:

∀ P & DNF / [ ] P$DNF,

i. If P = 'li' $\Longrightarrow$ D = {lo/l*C*} ≡ ':a' (D) $\rightarrow\!\!\!\!\rightarrow$ Ø.

Rule *i.* is motivated by a phonotactic elision of ':a' after 'li'.

ii. If P = {bi/ka} $\Longrightarrow$ D = {:alo/:al*C*}.

Rule *a.*, together with rule II.1.6.1.*c.*, in effect means that the Determiner ':alo', "the", now has four different allomorphs: {:alo, :al*C*, lo, l*C*}.

b. The following CASE GOVERNMENT condition means that Bound Prepositions can be attached only to a NF which carries an oblique CASE MARK or to a NF which is in one of the exceptional categories, as follows:

∀ P & CNF / [ ] P$CNF $\Longrightarrow$

i. If CNF [+acm], then value (CASE MARK (CNF)) $\rightarrow\!\!\!\!\rightarrow$ [+obm], where CNF ≠ DNF/ANF, and CNF is a Diptote of Type 1.

ii. If CNF [+acm], then value (CASE MARK (CNF)) $\rightarrow\!\!\!\!\rightarrow$ [+ncp], where CNF is a Numeral in the singular.

iii. Else, CNF [+obm]/CNF [+ncp].

The reason for rule *i.* is that Diptote Nominals of Type 1 are inhibited—when they are not definitized—from carrying oblique CASE MARKS. Hence, 'bimana|zila', "at houses", is acceptable with an accusative mark, but the value of this mark has to be MODIFIed to oblique; however, '*bimana|zili' is not acceptable, while 'bi:alomana|zili', "at the houses", is legal, since the inhibition is lifted upon definitization.

This curious phenomenon is often explained by the risk of confusing '*mana|zili', "*houses", with 'mana|ziliy', "my houses", as they sound similar. The CASE MODIFICation to Diptotes of Type 1 ensures that they will agree only with oblique Adjectives. Note the legal *expression*: 'bimana|zila kabiyrati+', "at big houses"—consisting of an accusative Nominal, and an oblique Adjective—but the unacceptable '*bimana|zila kabiyrata+', consisting of a Nominal and an Adjective both having accusative CASE MARKS, with the Nominal in fact being in oblique deep CASE.

The reason for rule *ii.* is that Numerals in the singular can acquire a *state* of *temporary declinability* by taking a surface accusative CASE MARK, which is *frozen*, or *neutralized*, in deep CASE significance, viz., it becomes neutral of CASE. Hence, we notice some curious rules

of Numeral-Nominal agreement in Arabic. Thus, both 'lixamosi bana|ti+', "for five girls"—with oblique Numeral—and 'lixamosa ɛasvarata binota+', "for fifteen girls", with two Numerals carrying a neutralized accusative CASE MARK, are acceptable.

The neutralization of such CASE MARKS allows Numerals like ':ih-oday', "one", to be used in either oblique or accusative CASE. Thus, note that both '*lixamosa bana|ti+' and '*lixamosi ɛasvarata binota+' are unacceptable, while both 'li:ih-oday :alobana|ti', "for one of the girls", and 'li:ih-oday ɛasvarata binota+', "for eleven of the girls", are acceptable.

Rule *iii.* accepts all other Nominals, including Numerals, after a Bound Preposition, provided the CNF carries a neutral or oblique CASE MARK. Thus, rule *iii.* also implies a constraint on P$CNF sequences such that:

iv. *P$CNF where CNF [+nmm] or where CNF [+acm] not in *i.* or *ii.*

II.2 HOMONYMY

In Chapter 4, § II.2, we alluded to the vital importance of *shape* or *how things look* in relation to the identification of these things by automatic or other procedures. In studying the structure of Nominals in this Chapter, we have so far expressed generalized statements and rules about patterns, stems, affixes, Simple and Complex NFs. However, we have also restricted our generalizations by formulating affixation conditions such as CASE GOVERNMENT, graphotactic, and compatibility rules. Another restrictive aspect to the above generalizations is homonymy: several of the Nominal structures listed exhibit ambiguities in NUMBER, GENDER, CASE, FLEXION, DEFINITENESS, REFERENCE, HUMANITY, and CATEGORY. Stems and patterns display a plurality of use which makes them difficult to distinguish on the basis of structure alone. Further, several morphological forms such as affixes are common to both Verbs and Nominals. Fortunately, in most cases, we can specify structural contexts in which disambiguation can be performed.

II.2.1 Stem Ambiguity across Pattern Distribution and Derivational Types

In Lists 1 to 6 (App. A, § B), we divided patterns into three derivational groups: MF, MO, and FO. However, we notice that several of the Nominal stems in those Lists, belong in more than one of the groups. In each group, a given stem has one or more Base patterns, each of which is ASSOCIATed with a number of possibly different BP-patterns. For example, the stem 'ld', "son, birth, ...", has the BASE MO-Pattern 'wacac', ASSOCIATed with the BP-Patterns ':awoca|c, wicocat'. These patterns serve to derive the Nominals 'walad', "son",

and ':awola|d, wilodat', "sons". This stem also has the Base MO-Pattern 'waciyc', ASSOCIATEd with the BP-Pattern 'wicoca|n'. These patterns are used to derive 'waliyd, wiloda|n', "son, sons". Further, the stem 'ld' has the Base FO-Pattern 'waciyc'—ASSOCIATEd with the BP-Pattern 'waca|:ic'—as well as the Base FO-Pattern 'wica|c', with no ASSOCIATEd BP-Patterns. These are used, respectively, to generate 'waliydat, wala|:id', "daughter, daughters", and 'wila|dat', "birth".

The plurality of these patterns ASSOCIATEd with the same stem in order to generate Nominals, which are—to varying degrees—semantically related, is parallel to the plurality of CATEGORIES of pattern that a given stem might have, in that each stem has derivational patterns for Verb, Verbals, and possibly Substantive, Adjective, and Numeral CATEGORIES. This plurality is a source of ambiguity, since the stem stays the same in form. The distinction of derivational groups or types, as well as the division into categorial classes, help us to disambiguate between kinds of pattern for each stem. Table 47 below gives a few examples of ambiguity in the distribution of patterns for stems.

PATTERN NUMBER & LIST	STEM	PATTERN	BROKEN PLURAL PATTERN	DERIVATIONAL TYPE
7 (3)	bn	cica|:	:acociyat	(M, F)
40 (3)		:icoc	:acoca|:	(M)
64 (3)		cic	N/A.	(F)
65 (3)		:icoc	N/A.	(F)
6 (3)	ktb	caciyc	caca|:ic	(M, F)
28 (3)		cica|c	cucuc	(M)
57 (3)		cica|c	N/A.	(F)
38 (3)	ld	wacac	:awoca|c, wicocat	(M)
39 (3)		waciyc	wicoca|n	(M)
66 (3)		waciyc	waca|:ic	(F)
67 (3)		wica|c	N/A.	(F)
17 (3)	h-sb	cica|c	cucoca|n	(M)
18 (3)		cacac	:acoca|c	(M)
1 (4)	drs	macocac	maɛa|cic	(M, F)
7 (4)		micocac	maɛa|cic	(M)

TABLE 47: Stem Ambiguity across Pattern Distribution and Derivational Types

II.2.2 Derivational Pattern Homonymy across Categorial and NUMBER/GENDER Values

We hinted, in § II.2.1 above, that stems have derivational patterns for several CATEGORIES of Nominal. In some cases, the same pattern may be used to generate a different CATEGORY from its normal use. For example, the pattern 'ca|cic' is normally associated with the Active Verbal CATEGORY. However, this pattern can be used to generate 'sa|ɛid', "forearm", which is a Substantive. In other cases, the same pattern can be used to generate different NUMBER categories, in the same derivational CATEGORY. For instance, 'cayoc' is used to generate 'bayod;at', "one egg", and 'bayod;', "many eggs". The use of the suffix 'at' can help us to disambiguate in such cases. We have already noted how the pattern 'macocac' can be used for "Locative of Time" or "of Place". Since the distinction is purely semantic, no structural (morphological) values or properties can be found to disambiguate such pattern homonymy.

Table 48 below gives some examples of homonymy in the distribution of patterns across Nominal and NUMBER/GENDER categories.

II.2.3 Suffix Homonymy across NGP, TENSE, MOOD, CASE, and FLEXION Values

In Tables 39 to 41, we summarized the NGC and FLEXION values of Nominal suffixes. However, the distribution of these properties is not clear cut, in that there is not a unique one-to-one correspondence between each suffix and possible linguistic properties. Each suffix is initially ASSOCIATed with NGC and FLEXION values, but may simultaneously be used for other alternative values.

Further, Nominal suffixes can be divided into five groups:

1) Purely Vocalic or CASE suffixes: {u, u+, a, a+, i, i+};

2) Purely Feminine GENDER suffixes: {at, ot, a|:, a|t};

3) Purely Defective-Nominal suffixes: {iy, ay, a|:};

4) Sound and Defective NGC suffixes: {iy, uw, a|, a|ni, ayoni, ayo, awo, uwna, awona, iyna, ayona}; and finally,

5) Supernumerary suffixes, which are inserted for graphotactic reasons: {iy, uw, a|, ah, aw}, as in ':axiyhi', "his brother", and ':axawayohi', "his two brothers".

Membership of some of the suffixes in several of the above categories complicates the selection of unique property values for the Nominals to which they are affixed. Hence, the importance of distinguishing stem and pattern types and CATEGORIES in disambiguating some homonymic values. In addition, most Nominal suffixes are shared with Verbs, at which

PATTERN NUMBER & LIST	PATTERN	TYPICAL SUFFIX	CATEGORY & NG VALUES	NOMINAL TYPE & STEM
1 (1)	ca|cic		L1	So.
3 (3)			NN	So.
7 (1)	wa|cic		L1	Df.
6 (6)			MM	Df.
10 (3)	caciyc		NN	So.
8 (5)			AA	So.
39 (3)	waciyc		NN	Df.
18 (5)			AA	Df.
15 (1)	muca|c	iy	L1	Df.Wk.
10 (2)		ay	L2	Df.Ab.
25 (1)	mucacc	iy	L1	Df.Wk.
12 (2)		ay	L2	Df.Ab.
59 (3)	cacac	Ø	NN, (F) BP	So.: {h-dq, qs;r}.
59 (3)		at	NN, (F) S	So.: {h-dq, qs;r}.
69 (3)	cayoc	Ø	NN, (F) BP	Df.: {bd;}.
69 (3)		at	NN, (F) S	Df.: {bd;}.
15 (5)	cucoc	Ø	AA, (M, F) BP	So.: {h-mr, xd;r, zrq, s;fr}.
20 (5)	cuwc	Ø	AA, (M, F) BP	Df.: {sd}.
21 (5)	ciyc	Ø	AA, (M, F) BP	Df.: {bd;}.

TABLE 48: Derivational Pattern Homonymy across Nominal Types and CATEGORIES

time, the suffix will have different NGP, TENSE, and MOOD values. However, there is one property which has non-homonymic values for Nominals, and that is the property PERSON. Thus, for all Nominals, except Pronouns, the ASSOCIATed default PERSON value is third.

However, we have found that the most complicated suffix in the whole corpus of analysed data, and the one with the highest number of homonymic property values, is the suffix 'iy'. In addition to its use in Verb structures, this suffix can be used in Nominal structures as a MRP suffix, as a Df.N ending, or as a Supernumerary suffix. Finally, 'iy' is also a Genitive Pronoun with the values: 1 (M, F) S, obm, dfp, col, GP.

Table 49 below summarizes the possible NGP, TENSE, MOOD, CASE, and FLEXION values for Nominal suffixes.

SUFFIX	(N) NGC & FLEXION VALUES	(V) NGP, TENSE & MOOD VALUES
u	(M, F) S, BP & (F) RP in NOM, svm.	1 (M, F) S, D, P, 2 (M) S & 3 (M, F) S in I, J.
u+	(M, F) S, BP & (F) RP in NOM, fvm.	N/A.
a	(M, F) S, BP in ACC & OBL, svm.	1 (M, F) S, D, P, 2 (M) S & 3 (M, F) S in S, J. 3 (M) S, prf.
a+	(M, F) S, BP in ACC, fvm.	N/A.
i	(M, F) S, BP in OBL & (F) RP in ACC & OBL, svm.	1 (M, F) S, D, P, 2 (M) S & 3 (M, F) S in J.
i+	(M, F) S, BP in NOM & OBL & (F) RP in ACC & OBL, fvm.	N/A.
at	(M, F) BP & (F) S.	N/A.
ot	(F) S.	N/A.
a|t	(F) S, RP & (M) BP.	N/A.
a|:	(M, F) S, BP.	N/A.
ay	(M, F) S & (M) BP, NEUT, xvm.	1 (M, F) S, D, P, 2 (M) S & 3 (M, F) S in I, S. 3 (M) S, prf.
ah	(F) S, D, RP.	N/A.
aw	(M) D.	N/A.
iy	(M, F) S, BP in NOM & OBL, svm; (M) RP in ACC & OBL, dnf & (M) S in OBL, svm.	1 (M, F) S, D, P, 2 (M) S & 3 (M, F) S in I; 2 (F) S in S, J.
uw	(M) RP in NOM, dnf & (M) S in NOM, svm.	1 (M, F) S, D, P, 2 (M) S & 3 (M, F) S in I.
a|	(M, F) D in NOM, dnf & (M) S in ACC, svm.	2, 3 (M, F) D in S, J & 3 (M) S, D, prf.
a|ni	(M, F) D in NOM, nnf.	2, 3 (M, F) D in I.
ayo	(M, F) D in ACC & OBL & (M) RP in ACC & OBL, dnf.	2 (F) S in S, J.
ayoni	(M, F) D in ACC & OBL, nnf.	N/A.
awo	(M) RP in NOM, dnf.	N/A.
uwna	(M) RP in NOM, nnf.	2, 3 (M) P in I.
awona	(M) RP in NOM, nnf.	2, 3 (M) P in I.
iyna	(M) RP in ACC & OBL, nnf.	2 (F) S in I.
ayona	(M) RP in ACC & OBL, nnf.	2 (F) S in I & 2 (F) P in I, S, J.

TABLE 49: Nominal Suffix Homonymy across NGP, TENSE, MOOD, CASE, and FLEXION Value

II.2.4 The Disambiguation of Nominal Forms Ending in 'iy'

We noted, in § II.2.3 above, that the most complicated structures to analyse and identify are those containing the suffix 'iy'. We imputed this complexity to the homonymy of property values ASSOCIATed with this suffix and to the multiplicity of functions or uses that it may have, namely, as plural, Defective, Supernumerary, Genitive, or Verb suffix.

We have already specified in Chapter 4, the conditions under which disambiguation can be

performed for V-Structures with 'iy'. However, we now need to specify the conditions for the disambiguation of Nominals ending with 'iy'. There are, in fact, three types of NFs that should be considered here: first, singular-plural Df.Wk.Ns; second, MRP So.Ns; and third, singular, FRP, and BP So.Ns. There is no homonymy in the third group, since 'iy', in this case, is either patently a Genitive Pronoun or patently a Supernumerary suffix.

In the second group, the ambiguity involved is between plural or Genitive Pronoun ending. Hence, the NF can be a human indefinite plural Modifier—in the accusative or oblique CASE and with reduced NGC suffix—or it can be a definite singular collocutive Annexed Nominal in neutral CASE and with a single vowel mark value for FLEXION.

In the first group, the ambiguity involved is between plural, Defective, or Genitive ending. This means that the NF in question is often indeterminate in NUMBER. This indeterminacy in NUMBER also implies an indeterminacy in CASE, FLEXION, DEFINITENESS, REFERENCE, CATEGORY, and HUMANITY. We will explain (in Ch. 12) how the property HUMANITY relates to the rules of agreement in Arabic.

Some of this homonymy can be resolved using affix context. To take an example: the NF Modifier 'ba|niy' could be interpreted as "(the) builder (of)" or "(the) builders (of)". The first interpretation means that the singular NF is either in the nominative or oblique CASE, with a single vowel mark value for FLEXION; and the second interpretation means that the human MRP NF is either in the accusative or oblique CASE, with reduced NGC suffix.

The disambiguation of these values can be made using the affixation conditions specified above. For example, the suffixes: {a, a+} do not occur with RPs; hence, both 'ba|niya', "(the) builder (of)", and 'ba|niya+', "a builder", can only be in the singular. The Determiner ':alo', "the", cannot co-occur with the reduced NGC suffix of MRPs; hence ':aloba|niy', "the builder", again, can only be in the singular. Bound Prepositions can only occur in Prepositional NFs; hence 'liba|niy', "to (the) builder(s) (of)", can only be interpreted as having an oblique CASE.

Table 50 below (where § *a.* applies to Singular-Plural Defective Weak Nominals, § *b.* to MRP Sound Nominals, and § *c.* to Singular, FRP, and BP Sound Nominals) illustrates in detail the various contexts under which the different property values, for NFs ending in 'iy', can be disambiguated.

II.2.5 Simple and Complex NF and Other Types of Categorial Homonymy

We discussed, in § II.2.1 and § II.2.2, certain types of categorial homonymy induced by the multiplicity of patterns for the same stem or by the multiplicity of stems with the same pattern. When this derivational pattern homonymy coincides with homonymy in suffix form, the

	(N) CONFIGURATION	NGC & FLEXION VALUES	DFN	HTY	CAT	EXAMPLE
a.1	NF$'iy'	(M) S in NOM & OBL, svm.	ndf	Ø	MM	't-ama|niy' "eight"
		(M, F) BP in NOM & OBL, svm.	ndf	hm, Ø	MD	'bawa|niy' "builders"
		(M) S, RP in NOM & OBL, svm-dnf.	ndf	hm/Ø	MD	'ba|niy' "builder(s)"
2	NF$'iy'$<a>	(M) S in ACC, svm.	ndf	Ø	MM	't-ama|niya' "eight"
		(M, F) BP in ACC & OBL, svm-fvm.	ndf	hm, Ø	XN	'bawa|niya' "builders"
		(M) S in ACC, svm.	ndf	Ø	MD	'ba|niya' "builder"
3	NF$'iy'$<a+>	(M) S in ACC, fvm.	ndf	Ø	MM	't-ama|niya+' "eight"
		(M) S in ACC, fvm.	ndf	hp	L1	'ba|niya+' "a builder"
4	NF$<i+>	(M) S in NOM & OBL, fvm.	ndf	Ø	MM	't-ama|ni+' "eight"
		(M, F) BP in NOM & OBL, fvm.	ndf	hm, Ø	NN	'bawa|ni+' "builders"
		(M) S in NOM & OBL, fvm.	ndf	hp	L1	'ba|ni+' "a builder"
5	NF$'iy'$GP	(M, F) BP in NEUT, svm.	dfp	hm, Ø	AN	'bawa|niyhi' "his builders"
		(M) S, RP in NEUT & svm.	dfp	hm/Ø	AN	'ba|niyhi' "his builder(s)"
6	PNF'iy'	(M) S in OBL, svm.	ndf	Ø	PMM	'lit-ama|niy' "to eight"
		(M, F) BP in OBL, svm.	ndf	hm, Ø	PMD	'libawa|niy' "to builders"
		(M) S, RP in OBL, svm-dnf.	ndf	hm/Ø	PMD	'liba|niy' "to builder(s)"
7	DNF'iy'	(M) S in NOM & OBL, svm.	dfp	Ø	DM	':alt-t-ama|niy' "the eight"
		(M, F) BP in NOM & OBL, svm.	dfp	hm, Ø	DN	':alobawa|niy' "the builders"
		(M) S in NOM & OBL, svm.	dfp	Ø	DN	':aloba|niy' "the builder"
8	PDNF$'iy'	(M) S in OBL, svm.	dfp	Ø	PDM	'lilt-t-ama|niy' "to the eight"
		(M, F) BP in OBL, svm.	dfp	hm, Ø	PDN	'lilobawa|wa|niy' "to the builders"

TABLE 50: Disambiguation Contexts for Nominal Forms Ending in 'iy'

		(M) S in OBL, svm.	dfp	Ø	PDN	'liloba|niy' "to the builder"
9	PNF'iy'$GP	(M, F) BP in OBL, svm.	dfp	hm, Ø	PAN	'libawa|niyhi' "to his builders"
		(M) S, RP in OBL, svm.	dfp	hm/Ø	PAN	'liba|niyhi' "to his builder(s)"
b.1	NF$'iy'	(M) S in NEUT, svm.	dfp	Ø	AN	'mudarrisiy' "my teacher"
		(M) RP in ACC & OBL, dnf.	ndf	hm	MD	'mudarrisiy' "my teachers"
2	NF$'iy'$'ya'	(M) RP in NEUT, svm.	dfp	hm	AN	'mudarrisiyya' "my teachers"
3	NF$'iy'$GP	(M) RP in ACC & OBL, svm.	dfp	hm	AN	'mudarrisiyhi' "his teachers"
4	PNF'iy'	(M) S in OBL, svm.	dfp	Ø	PAN	'limudarrisiy' "to my teacher"
		(M) RP in OBL, dnf.	ndf	hm	PMD	'limudarrisiy' "to teachers"
5	PNF'iy'$'ya'	(M) RP in OBL, svm.	dfp	hm	PAN	'limudarrisiyya' "to my teachers"
6	PNF'iy'$GP	(M) RP in OBL, svm.	dfp	hm	PAN	'limudarrisiyhi' "to his teachers"
c.1	NF$'iy'	(M) S in NEUT, svm.	dfp	Ø	AN	'kita|biy' "my book"
		(F) S in NEUT, svm.	dfp	Ø	AN	':uxotiy' "my sister"
		(F) RP in NEUT, svm.	dfp	hm	AN	'mudarrisa|tiy' "my teachers"
		(M) BP in NEUT, svm.	dfp	Ø	AN	'kutubiy' "my books"
2	NF$'iy'$GP	(M) S in OBL, svm.	dfp	hm	PAN	':abiyhi' "his father"
3	PNF'iy'	(M) S in OBL, svm.	dfp	Ø	PAN	'likita|biy' "to my book"
		(F) S in OBL, svm.	dfp	Ø	PAN	'li:uxotiy' "to my sister"
		(F) RP in OBL, svm.	dfp	hm	PAN	'limudar risa|tiy' "to my teachers"
		(M) BP in OBL, svm.	dfp	Ø	PAN	'likutubiy' "to my books"
4	PNF'iy'$GP	(M) S in OBL, svm.	dfp	Ø	PAN	'li:abiyhi' "to his father"

TABLE 50: Disambiguation Contexts for Nominal Forms Ending in 'iy' (Contd)

ambiguity can be compounded. This is further complicated upon the affixation of clitic Pronouns to such homonymic NFs.

We have already discussed, in detail, the homonymy of Simple CFs with Simple NFs, in Chapter 4, § II.2.6, and of Complex CFs with Complex NFs in Chapter 4, § II.4.2; together with disambiguation techniques. Note also, that all the homonymic cases of SCFs and CCFs—listed in Tables 20 and 24—can also be disambiguated, using the affixation conditions of Determiners and Prepositions to other forms:

Given XF as a homonymic Form,

a. DXF can be interpreted only as DNF, since *DCF, where CF is a Verb.

b. PXF can only be PNF—where NF is in the oblique CASE—or PCF, where CF is in the imperfect TENSE and subjunctive MOOD. The homonymy described here arises in the case of 'li', "to, for, ...", which can be a Particle governing Verbs in the subjunctive MOOD or a Bound Preposition governing Nominals in the oblique CASE. This constraint applies to all Nominals as specified in II.1.7.*b*.

c. Otherwise, PXF can only be PCF, where P = 'sa', and PNF, where P = {bi, ka, la}.

In addition, the bound morpheme 'ka' can be a proclitic Bound Preposition—with oblique Nominal GOVERNMENT—or an enclitic Pronoun, with the values: 2 (M) S, dfp, col, cpm. We have also specified in Chapter 4, § II.4.1, the conditions for Subject/Accusative/Genitive Pronoun homonymy and disambiguation. Another type of homonymy arises, for example, when the Verbal pattern 'ca|cic' is used with the root 'h-sb', "count, think", in order to generate 'ha|sib', "counting/thinking", with homonymic TRANSITIVITY values, viz., either monotransitive or xtransitive.

A special kind of homonymy arises from the use of Supernumerary characters from the set: {s, :, l, t, m, w, n, i, y, h, |}, in Base and especially BP forms. For instance, in ':agoniya|:', "rich", both instances of <:> are Supernumerary; however, in ':l', "being first", and in 'mr:', "being feminine", the <:> is a radical, i.e., an integral part of the stem. Further, the prefix <m> can be a Verbal prefix, in 'mudarris', "teacher"; a Tlocative prefix in 'makotab', "desk"; or an M-Infinitive prefix in 'mad-ohab', "direction, ideology, etc".

II.3 ANNOTATED MS RULES FOR THE COMPLEX NF

We have, so far, provided a linguistic description of the Nominal in M.S.A.; and—in a formal account of this Nominal—we have formulated generalized morphological statements and rules constrained by CASE GOVERNMENT and Graphotactic Conditions and by exceptional instances of homonymy. From this body of rules and conditions, we can deduce formal

Morphology Structural Rules, or MS rules, in order to generate M-Markers for Complex Nominal Forms, such as those under investigation.

We stated, in § II.1.4 and § II.1.7, that a Simple NF is the realization of a shallow pattern, that is filled with a discontinuous lexical root, or stem—which occurs at the edges of syllables within that realization—and that attaches with an optional GENDER suffix (G), and with an obligatory CASE or NGC suffix (K).

We also stated, in § II.1.7, that a Complex NF is either a DNF—which is a Simple NF with a proclitic Determiner (D)—or an ANF—which is a Simple NF with an enclitic Genitive Pronoun (GP)—and that such Complex NFs can, in turn, be attached with a Bound Preposition (P). We will call these latter NFs *Prepositional NFs* (*PNFs*).

We have argued, in Ch. 4, § II.5.1, that such observations imply a formal morphological grammar which we refer to as *MS rules*. We have also argued (in Ch. 4, § II.5.2), that if such rules are formulated as a CFG, they are too powerful and unrestricted to generate "all and only" the grammatical morphological sequences of Arabic. They will not take account of the CASE GOVERNMENT, Graphotactic Conditions, and other rules that constrain the generation of M-Structures. They would also overgenerate by allowing invalid sequences of the type disallowed in § II.1.6 and § II.1.7 above. For instance, there is a structural constraint that we can deduce from § II.1.6: *DNFGP. In § II.1.6.1, we described FLEXION and DEFINITENESS conditions, which disallow *D$NF, where NF [+def], or NF [+fvm], or NF [+dnf] and NF $\neq$ MM/Df.

In § II.1.6.2, we described FLEXION and TRANSITIVITY conditions, which disallow *NF$GP, where NF is a Verbal and [+ntr], or where NF [+fvm]/NF [+nnf]. There was a CASE GOVERNMENT condition in § II.1.7 disallowing *P$CNF, where CNF [+nmm]/NF [+acm] and CNF is not a Diptote 1 or a singular MM. We also specified, in § II.1.6, GP and Determiner selection rules, as well as Graphotactic Conditions, which regulate the juxtaposition of affixes to NFs. We have referred to these conditions collectively as *Affix Selection Rules*, or *ASRs*.

The Graphotactic Conditions consist in Vocalic Compatibility (VCo) and simple boundary transformations such as:

- $\forall$ NFKGP $\Longrightarrow$ K $\longrightarrow$ Ø, where GP = 1 (M, F) S (cf. § II.1.6.2); and
- $\forall$ PDNF $\Longrightarrow$ ':a' $\longrightarrow$ Ø, where P = 'li' (cf. § II.1.7).

Finally, there were ASRs restricting the juxtaposition of G to K, where G is singular and K is plural; or where G is plural and K non-singular or accusative. Hence, the constraint expressed in II.1.4.*d.*:

*G$K, where G [+sn] and K [+pl]; or G [+pl] and K $\neq$ [+sn]/K [+acm].

ASRs—which consist in DEFINITENESS, FLEXION, TRANSITIVITY, CASE GOVERNMENT, and Graphotactic Conditions (GCs), as well as Vocalic Compatibility (VCo) conditions—inhibit the generation of sequences that would otherwise violate the morphological grammar well-formedness rules of Arabic.

In order to constrain any given CFG for Nominal sequence generation, and to generate "all and only" valid sequences, all the above conditions and constraints need to be incorporated in the CFG. Here again, we can augment CFGs to CSGs by inserting feature annotations into the structural rules as in *a.* below.

a.i. PNF ⟶ (P [+VCo, +GVT]) DNF [+CASE]
a.ii. PNF ⟶ (P [+GVT]) ANF [+CASE]
a.iii. DNF ⟶ (D [+VCo, +ASRs]) NF [+ASRs]
a.iv. ANF ⟶ NF [+VCo, +GCs] (GP [+VCo, +GCs])
a.v. NF ⟶ P/S [+GCs] (G [+ASRs]) K [+ASRs]
a.vi. P/S ⟶ pattern ... stem ... pattern ... stem [+ASRs]
a.vii. P ⟶ {li/bi/ka} [+VCo, +OBL]
a.viii. D ⟶ {:alo/:alC/lo/lC} [+VCo]
a.ix. G ⟶ {at/ot/a|t} [+GCs, +ASRs]
a.x. K ⟶ {uw/uwna/iy/iyna/ayo/ayoni/a|/a|ni/awo/awona/u/u+/a/a+/i/i+} [+GCs, +ASRs]
a.xi. GP ⟶ {iy/ya/na|/ka/ki/kuma|/kumo/kunna/hu/hi/ha|/huma|/hima|/humo/himo/hunna/hinna} [+VCo, +ASRs]

Note that, owing to lack of space, we do not expand the above annotations any further. We have already explained the conditions they represent. Grammar *a.* is specified in order to give an idea of what an adequate morphological CFG for Arabic should look like. This grammar should be seen as providing for optional and obligatory elements in a Nominal sequence, as follows:

Rules *a.i.* and *a.ii.* allow a sequence P$CNF, $\mathcal{IFF}$ the CASE of the CNF is equal to the CASE GOVERNMENT of P. Rules *a.i.* and *a.iii.* specify that P and D are to satisfy the VCo conditions. In rule *a.iii.*, D and NF also have to satisfy ASRs. In *a.iv.*, the sequence NF$GP is generated subject to VCo and GCs and, in *a.v.*, the sequence G$K has to observe GCs and ASRs. These rules (*a.i.* to *a.vi.*) are non-terminal rules which specify feature labels.

In contrast, the terminal rules in *a.vii.* to *a.xi.*, require that each lexical item specify its own particular GCs, VCo, ASRs, and obligatory GOVERNMENT-feature values. For example, in *a.vii.*, 'li', "to, for ...", would specify the deletion of ':a' in D, in a sequence PDNF, and so on. Using the grammar in *a.*, we can now generate Annotated M-Markers of the type shown in Figures 18–19 and we can generate Nominal sequences of the type in *b.* below:

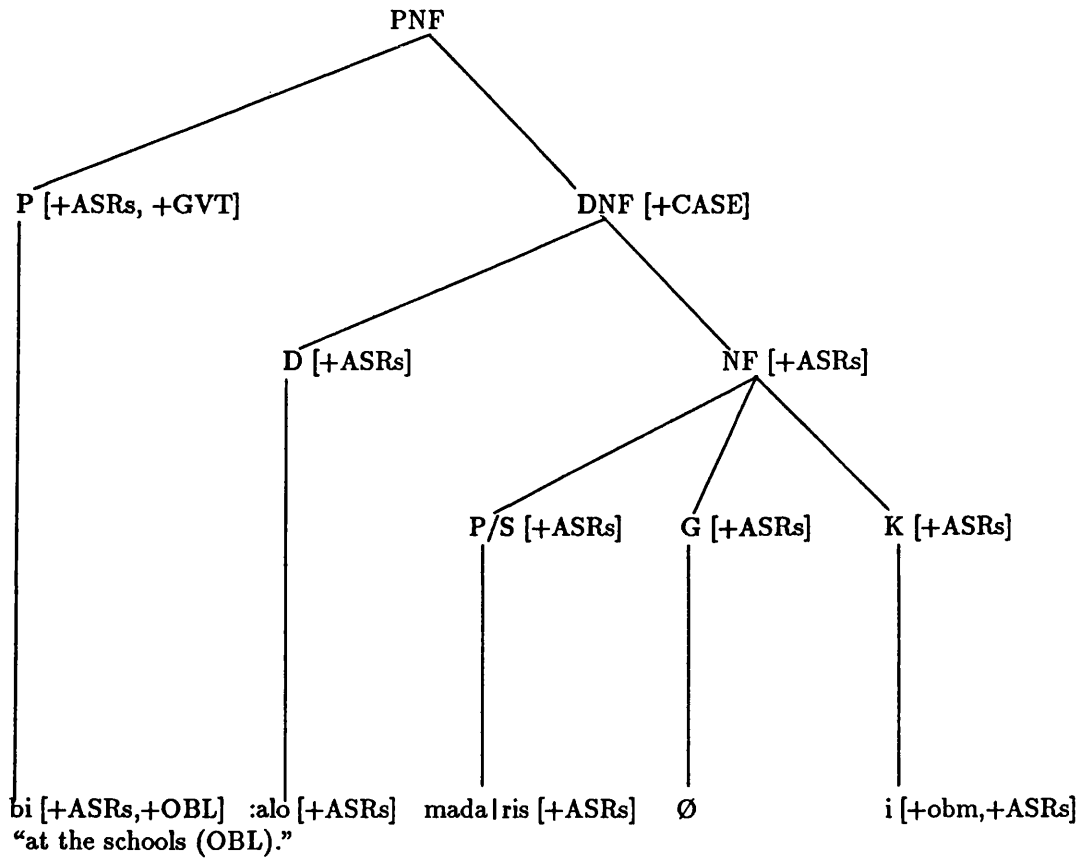


FIGURE 18: An Annotated M-Marker for a Complex NF of Type 1

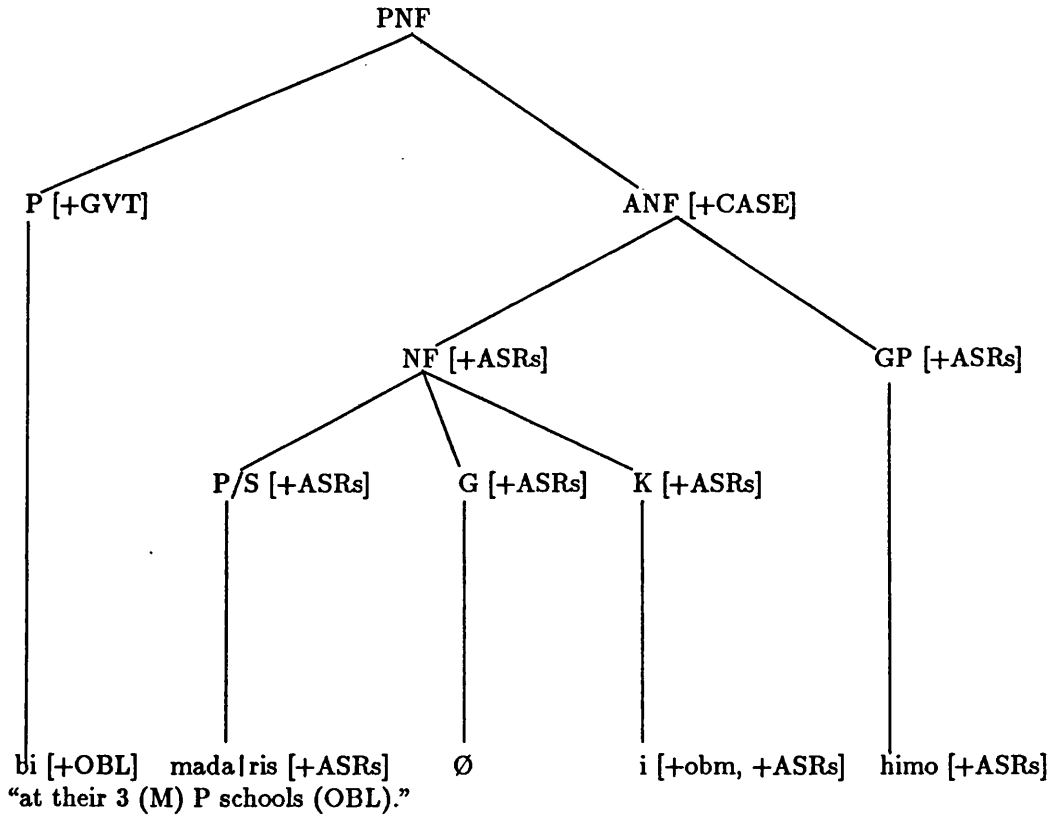


FIGURE 19: An Annotated M-Marker for a Complex NF of Type 2

b.i.	'madorasatu+', "a school":	NF
b.ii.	':alo$madorasatu', "the school":	D$NF
b.iii.	'madorasatu$hu', "his school":	NF$GP
b.iv.	'bi$madorasati+', "at a school":	P$NF
b.v.	'bi$:alomadorasati', "at the school":	P$DNF
b.vi.	'bi$madorasatihi', "at his school":	P$ANF
b.vii.	'madorasati', "(the) school (of)":	NF
b.viii.	'bi$madorasati', "at (the) school (of)":	P$NF
b.ix.	'ba|niy', "(the) builder(s) (of)":	NF
b.x.	':axuw$hu', "his brother":	NF$GP
b.xi.	'mabona|$hu', "his building":	NF$GP
b.xii.	'madorasat$iy', "my school":	NF$GP

==0== <*****> ==0==

Chapter 7

PROCESSING THE N-COMPLEX

INTRODUCTION

In Chapter 6, we gave structural descriptions for N-Complexes. In Chapter 7, we translate these descriptions into formal specifications which spell out for each Nominal morphological structure a description, a basic structure, LENGTH and boundary conditions, features and feature values, and a role.

From these defined structures we construct, in Section I.1, a set of automatic dictionaries and property lists that constitute a database for Nominals. From the formal specifications we are able to implement the Morphological Rules as a program called the N-Processor. This processor consists of several routines and procedures: there are initial closed subroutines that are defined independently in Chapter 3, for fast and primitive tasks such as SEGMENTation, word ASSEMBLY, MATCHing, and character PICKing. Other tasks include root EXTRACTion as defined in Chapter 5—but here as applied to Verbals—and stem EXTRACTion which is applied to non-Verbal Nominals; there are also intermediate closed subroutines responsible for Nominal DERIVations, and there are the main N-Recognition procedures. These are operations that carry out iterations or procedural instructions and, after identifying inputwords and their constituent morphemes, decide upon the morphological structural context. The identification of such contexts triggers the ASSIGNment of unique properties, and possibly the disambiguation or the flagging of homonymic structures.

There are other types of procedures which are also context-sensitive but do not perform any

explicit operations. Instead they control the output of lower routines by filtering out sequences that violate the Morphological Rules.

Section II synthesizes the linguistic structure of Chapter 6 and the computational structure of Chapter 7. It also describes the architecture and the organization of the N-Processor and outlines the search strategy in a Nominal parse. To illustrate such parses, there are examples of the performance of the N-Processor, as well as samples of output and time taken in a given N-Parse, which are provided in Appendix D, Section B.

I CONTENTS OF THE N-COMPONENT

I.1 THE N-DATABASE

I.1.1 Data Representation, Data Access, and Property ASSIGNment

In Chapter 3, we stated that Cambridge LISP provides system-defined facilities such as SETQ and FLUID to define simple data structures and to create dictionaries with identifiers and lexical entries, and with dynamic scope. We listed other facilities for data access, retrieval and manipulation such as MEMQ, LENGTH, LAST, EQUAL, XN, CAR, CDR, and CADR.

In Chapter 5, § I.1.1, we argued for the importance of an adequate representation method for morphological data in a given processor and specifically in the construction of a viable database with efficient access, storage, and data ASSOCIATions and manipulation. We stated, that the LISP functions described above facilitate such tasks; and we argued that the definition of dictionaries allows the formulation of dictionary-dependent rules or constraints such as TRANSITIVITY PROMOTion.

We also discussed the problem of complex data structures and the need for an adequate method of exclusive ASSOCIATion between given lexical entries such as roots and patterns, and argued for the use of PLIST structures as association lists with pattern lists as the values of radical arguments.

Following the same arguments, we can use all the facilities described above in the N-Processor. For instance, we can SET up a stem-pattern association structure as follows:

- (PUT pattern 'bn '(cica|: :icoc cic :acociyat :acoca|:)).

Thus, we can create a stem-pattern PLIST, or *PATLIST*. This PATLIST can be made more specific for more precise DERIVation and constrained access by distinguishing three types of pattern as in Chapter 6:

- (PUT masculine-feminine pattern 'bn '(cica|: :acociyat)).
- (PUT masculine pattern 'bn '(:icoc acoca|:)).
- (PUT feminine pattern 'bn '(cic)).

As we argued in Chapter 5, the retrieval of such structures can be achieved using the function ATSOC as in:

- (ATSOC 'stem (PLIST 'pattern)).

In Chapter 5, § I.1.2, we stated that, given the multiplicity of features and their non-unique or homonymic distribution, it is computationally complex and important, especially for Arabic, to solve the problem of property ASSIGNment: the ASSOCIATion of a *feature label* and *feature value* with a lexical entry. For the Nominal database, we also need to distinguish unique and homonymic properties and here again, we can use PUT, MAPC, and GET embedded in small and fast MACROS, in order to achieve the ASSIGNment and retrieval of unique properties. The problem of homonymy, as described in Chapter 6, § II.2, can also be solved in a parallel way to the solution of homonymy in Chapter 5. First, we note that the ASSIGNment of all properties, in the N-Database, follows the descriptions of Nominal properties as specified in Chapter 6. This ASSIGNment then is made as follows:

a. ASSIGN unique values to a lexical entry initially as default values. Then MODIFY initial values at processing time as fixed or as dynamic values. For example, the suffix 'a|', which is most commonly used for dual, initially carries in the database the value *nominative* for the property *CASE*. During processing, the following distinction is made:

- 'a|' / [ ] dual NF $\Longrightarrow$ 'a|' [+nominative].
- 'a|' / [ ] singular NF $\Longrightarrow$ 'a|' [+accusative].

The above values are ASSIGNed as unique and no further MODIFICation is necessary. The N-Database contains a number of unique values as defined in the V-Database (§ I.1.2) for NUMBER, GENDER, TRANSITIVITY, VOICE, and REFERENCE. In addition, the N-Database has these values:

PERSON:	third.
TENSE:	imperfect.
CASE:	nominative, accusative, oblique.
FLEXION:	full/single vowel mark, full/reduced NGC suffix.
TYPE:	Defective, bound, non-bound.
PLURAL CLASS:	Feminine Regular Plural, Masculine Regular Plural.

NUMERICAL VALUE:	1, 2, 3, 4, ...
HUMANITY:	human, non-human.
DEFINITENESS:	definite, indefinite.
CASE GOVERNMENT:	oblique.
CATEGORY:	MD, L1, L2, AA, MM, NN, DN, DM, DA, AN, AM, PL1, PL2, PNN, PMM, PAA, PDN, PDM, PDA, PAN, PAM, PMD.

b. ASSIGN neutral values to lexical entries with neutral values or with resolvable homonymy. Here again, the N-processor makes use of the same kind of *integrated*, or *neutral*, labels as the V-Processor such as *mf* for neutral GENDER, and *ncp* for neutral CASE. Here is a breakdown of neutral feature labels and values with some examples:

- NUMBER:
  - singular-plural: **sp**, e.g., 'iy'$Defective P/S as in 'ba|niy', "(the) builder(s) of".
- GENDER:
  - masculine-feminine: **mf**, e.g., ':aboniyat', "buildings".
- CASE:
  - nominative-accusative-oblique: **ncp**, e.g., 'ay'$Abbreviated P/S, as in 'mabonay', "building".
  - nominative-oblique: **npm**, e.g., <i+>$Defective P/S, as in 'ba|ni+', "a builder".
  - accusative-oblique: **cpm**, e.g., <i+> $FRP NF as in 'mudarrisa|ti+', "teachers".
- CATEGORY:
  - Modifier-Noun: **XN**, e.g., 'mabonay', "a building/(the) building (of)".
  - Modifier-Adjective: **XA**, e.g., ':uwlay', "first/first (of)".
- TRANSITIVITY: monotransitive-xtransitive: **mxtr**, e.g., 'h-a|sibu+', "thinking/ counting".

c. ASSIGN ambiguous values to entries that have homonymic values which may not be resolved by the morphological parser and will then need to be resolved by the syntactic analysis. Such cases are marked with an ambiguity *flag* starting with the prefix *x* for unknown. Most such cases are frequently correlated or concurrent. Here is a list of these:

i. The example of ':uwlay', "first/first (of)" is ambiguous for CATEGORY as above, and for FLEXION between: full and single vowel mark: **xvm**.

ii. The example of 'iy' as follows:

First, 'iy' / [ ] MRP L1/L2 is ambiguous for the properties and values:

NUMBER:	singular-plural: **sp**.
CASE:	accusative-oblique/neutral of CASE: **xxp**.
FLEXION:	single vowel mark/reduced NGC suffix.
HUMANITY:	human/non-human: **xh**.
DEFINITENESS:	definite/non-definite: **xdf**.
REFERENCE:	collocutive-exlocutive: **xxl**.
PLURAL CLASS:	MRP-singular: **rmx**.
CATEGORY:	Modifier-Annexed NF: **WN**.

Second, 'iy' / [ ] Defective Nominal is ambiguous for the following properties and values:

NUMBER:	singular-plural: **sp**.
CASE:	accusative-oblique/nominative-oblique: **xnp**.
FLEXION:	single vowel mark/reduced NGC suffix.
HUMANITY:	human/non-human: **xh**.
TYPE:	Defective/non-Defective.

d. ASSIGN two or more labels in the case of lexical entries with multiple pattern ASSOCIATions. This is due to the uniqueness of identifiers in LISP as explained in Chapter 5, § I.1.2. For example, for the stem 'ld', "child, birth ...", we can make the statements:

- (PUT 'mpattern1 'ld '(wacac :awoca|c wicocat)).
- (PUT 'mpattern2 'ld '(waciyc wicoca|n)).

e. There is another kind of pattern disambiguation which is performed in the N-Processor. This is the integration of certain homonymic suffixes in patterns in order to distinguish *productive* and non-productive or exceptional suffixes. For instance, the suffix 'at' in 'ka|tibat', "writer" is a *productive*, or *regularized*, feminine-singular suffix: 'ka|tibat' is generated from the pattern 'ca|cic' and the suffix 'at'; whereas, in 'katabat', "writers", 'at' is a non-productive, or non-regularized, suffix: 'katabat' is generated from the pattern 'cacacat' with the suffix being Supernumerary and integrated as part of the pattern. The first is a productive MATCHing-and-SEGMENTation process and the second is merely a MATCHing process. To give a similar example, 'iy' in 'mu:a|xiy', "being a brother", is the Defective suffix 'iy', and 'mu:a|xiy' is generated from 'muca|c' and 'iy'. However, in 'mu:a|xay', "having a brother", the Abbreviated suffix 'ay' is integrated as part of the pattern 'muca|c'. This technique helps to disambiguate the two different occurrences of the pattern 'muca|c' which in the first case generates,

by MATCHing and SEGMENTation, examples such as 'mu:a|xiy' and 'mu:a|xi+', "being a brother"—which are of the L1 CATEGORY—and in the second case generates, by MATCHing and simple transformation, examples such as 'mu:a|xay' and 'mu:a|xawona', "having a brother", which are of the L2 CATEGORY.

I.1.2 The Structure of the N-Database

Similarly to the V-Database, the N-Database is a superset consisting of:

- A set of dictionaries defined by the function SETQ and having fluid identifiers and specific values with dynamic scope. These dictionaries help to formulate dictionary-dependent rules.
- A set of properties, with initial and dynamic property values, and a set of association lists defined on the Nominal lexical entries, using the functions PUT and MAPC. Each lexical entry is thus defined as a MEMBER of a given dictionary and carries one or more PLISTS ASSOCIATed with it, and the set of dictionaries thus defined constitutes the N-Database. These are listed (in Vol. II, App. B, Sections B–F) as follows:

a. A dictionary of pattern ASSIGNments: the set of roots and stems each being ASSOCIATed with its characteristic patterns. These pattern lists are then ASSIGNed CATEGORY and VOICE values. The pattern lists are of six CATEGORIES:

i. Verbal1 and Verbal2 (The L-Database).

ii. Substantive, or Noun (The S-Database).

iii. Tlocative (The C-Database).

iv. Adjective (The A-Database).

v. Numeral (The M-Database).

For each CATEGORY the N-Database lists for each entry on a new line and from left to right:

- Entries in Masculine-Feminine pattern lists.
- Entries in Masculine pattern lists.
- Entries in Feminine pattern lists.

These pattern lists are merged into a number of Global Pattern Lists such as "*MLIST*" and "*NLIST+*"—for Masculine-only DERIVations—"*FLIST*" and "*!*FLIST*"—for Feminine-only DERIVations—and "*XLISTS*", for Masculine-Feminine and exceptional DERIVations.

The divisions are overlapping in that a given pattern list may belong in more than one Global List depending on its DERIVational constraints.

b. A set of N-Affix dictionaries as follows:

i. A dictionary of Nominal CASE-only suffixes.

ii. A dictionary of Nominal GENDER-only suffixes.

iii. A dictionary of Nominal NGC suffixes.

iv. A dictionary of Bound Prepositions.

v. A dictionary for the allomorphs of the Arabic Determiner.

vi. A dictionary of NUMERICAL VALUES where each Numeral stem is ASSOCIATed with a given NUMERICAL VALUE.

vii. A dictionary of N-PATLISTS.

Each of the above affixes carries its own initial values for NGC, DEFINITENESS, FLEXION, HUMANITY, CATEGORY, TYPE, REFERENCE, and PLURAL CLASS.

c. The N-Database, also makes use of the dictionary for Subject Pronouns, Demonstrative Pronouns and Proper Nouns as defined in the V-Database. Each of the Pronouns and Proper Nouns in that dictionary carries initial values for NGC, DEFINITENESS, FLEXION, HUMANITY, CATEGORY, and REFERENCE.

d. In addition, the N-Database makes use of the dictionary for V-Roots as defined in the V-Database in order to retrieve TRANSITIVITY values for Verbals only. It also makes use of the dictionary for enclitic Pronouns, used here as Genitive Pronouns; and of the feature PLISTS given in § I.1.2 above.

I.2 THE N-PROCESSOR

I.2.1 Initial Closed Subroutines

Similarly to the V-Processor, the N-Processor makes use of the independently-defined closed subroutines (in Ch. 3, § II). Hence, we will assume the following subroutines SEGM, SEGMV, MNSEGM, SCRIPT, MKWORD, MAKESTRING, NTH, NTHV, SUBST1, SUBST2, DELETEV, and DELETEW. We will also use the system-defined functions: CONS, DELETE, SUBSTITUTE, COPY, SETQ, PUT, MAPC, ATSOC, EXPLODE, REVERSE, MEMBER, LENGTH, and RETURN as described in Chapter 3, § I.4.

We will also assume a specification table for Nominal Patterns that is parallel to that given

in Table 1 for general patterns. Nominal Patterns are different only in two respects: 1) they may have lexical Supernumerary segments in the case of Broken Plural Pattern types; 2) they carry CATEGORY values such as L1, L2, NN, MM, AA, and MD, as well as VOICE values in the case of Verbals only. The role of N-Patterns is to constrain lexically and structurally the DERIVation or realization of NFs in conjunction with N-Stems. We will then assume pattern MATCHers: MATCHX and MATCHW as defined in Chapter 3, § II.3.

In addition, we will assume a specification table for Nominal stems that is parallel to that given in Table 25 for V-Roots. Nominal stems are different only in two respects: 1) only Verbal roots have Augmented forms and TRANSITIVITY values; 2) Numeral stems carry NUMERICAL VALUES such as 1, 2, 3, etc. The role of N-Stems is to constrain lexically the DERIVation of NFs in conjunction with N-Patterns. Given the similarities between V-Roots and Verbal Roots and the differences between the latter and N-Stems, as observed in Chapter 6, § I.1, we need two different EXTRACTion subroutines: one for Verbal roots and one for N-Stems. For the EXTRACTion of Verbal roots we will assume a root EXTRACTion subroutine: EXROOT as defined in Chapter 5, § I.2.1.3. STEPS IV, VII, and VIII, in EXROOT, serve to identify Verbal roots.

This identification is achieved by applying the following operations:

1) The DELETion of the initial Supernumerary character <m> from an Augmented Defective Hollow form or from an Augmented Sound form, and the ASSIGNment of the appropriate TRANSITIVITY value: intensive roots are given the same values as their Basic counterparts, and causative roots are PROMOTed to higher values than their Basic counterparts (in STEP IV and STEP VII).

2) The DELETion of the Supernumerary initial and final characters: {m, t}, and the ASSIGNment of TRANSITIVITY values as in (1) above (in STEP VII).

3) The DELETion of the medial Supernumerary character <:> which is transformed from an original <w> or <y> (in STEP VIII).

Having specified how EXROOT is to be used in order to identify Verbal roots, we now need to encode a basic subroutine to identify N-Stems. This identification is different from Verbal and Verb root EXTRACTion in two respects: there is no TRANSITIVITY ASSIGNment; and there is a larger number of Supernumerary characters to DELETE. We can define a subroutine: *EXSTEM*, for N-Stem *EXTRACT*ion, which takes two arguments, an inputword and a rootlist as provided in the Databases. Given the inputword, which is here a non-Verbal NF, EXSTEM needs to EXTRACT, from it, a purely consonantal form—which is the set of radicals—by performing predetermined simple linguistic transformations including DELETions of semivowels and Supernumerary characters. Thus, the operations of EXSTEM are procedural: they are

linguistic rules with the form: *if* case *n*, *then* apply rule *x*. The order of the cases means that a character that is an integral part of the stem but is homonymic with a Supernumerary character, does not in fact get DELETed. The overall operation of EXSTEM is constrained to the LENGTH interval: [2, 5], with 2 being the threshold, or the minimum LENGTH characteristic of Defective stems, and 5 being the upperbound, or maximum LENGTH characteristic of a raw stem with a maximum number of Supernumerary characters.

Figure 20 below shows a flowchart representation for the procedural algorithm for EXTRACTing stems: EXSTEM.

I.2.2 Intermediate Closed Subroutine: DERIVation

After defining a repertoire of basic subroutines for TRANSCRIPTION, SEGMENTation, word ASSEMBLY, MATCHing, root and stem EXTRACTion, and character PICKing, we now need to define an intermediate subroutine that makes use of that repertoire in order to obtain a Verbal or non-Verbal Nominal *DERIVation*. Further, we have distinguished two different types of derivational patterns: Basic and Broken Plural types which are subdivided into three groups: MF, M, and F.

In order to obtain a Nominal DERIVation, *NDERIVE* (which is the function we define for this purpose) proceeds as follows: first, it EXTRACTS a root (for Verbals) or a stem (for non-Verbals) from the inputword. Then, it scans Global Pattern Lists (GPLs) such as "MLIST+" or "!*FLIST". GPLs are specified by higher processes than N-DERIVation, and each GPL is a set of association lists made up of *Local Pattern Lists* (LPLs), each of which is a set of PAIRings of roots or stems and patterns. From such scans, NDERIVE is able to retrieve a pattern that MATCHes the inputword.

A successful DERIVation will RETURN the name of the LPL, the root or stem, and the pattern, all of which can be used as constraints on Nominal parsing tasks or as clues in disambiguating homonymic structures or properties. Nominal DERIVation is also constrained in that it can be made from a Basic pattern which resides in the front of the LPL, or from BP patterns which reside in the tail or REST of the LPL. Thus, we need four different N-DERIVation operations:

a. *DERIVEP*: to DERIVE Verbal roots from Basic patterns using EXROOT.

b. *DERIVEC*: to DERIVE non-Verbal stems from Basic patterns using EXSTEM.

c. *DERIVEQ*: to DERIVE Verbal roots from BP patterns using EXROOT.

d. *DERIVEZ*: to DERIVE non-Verbal stems from BP patterns using EXSTEM. For further simplification, we will describe DERIVEP and DERIVEC as one operation: *DERIVEA*, and

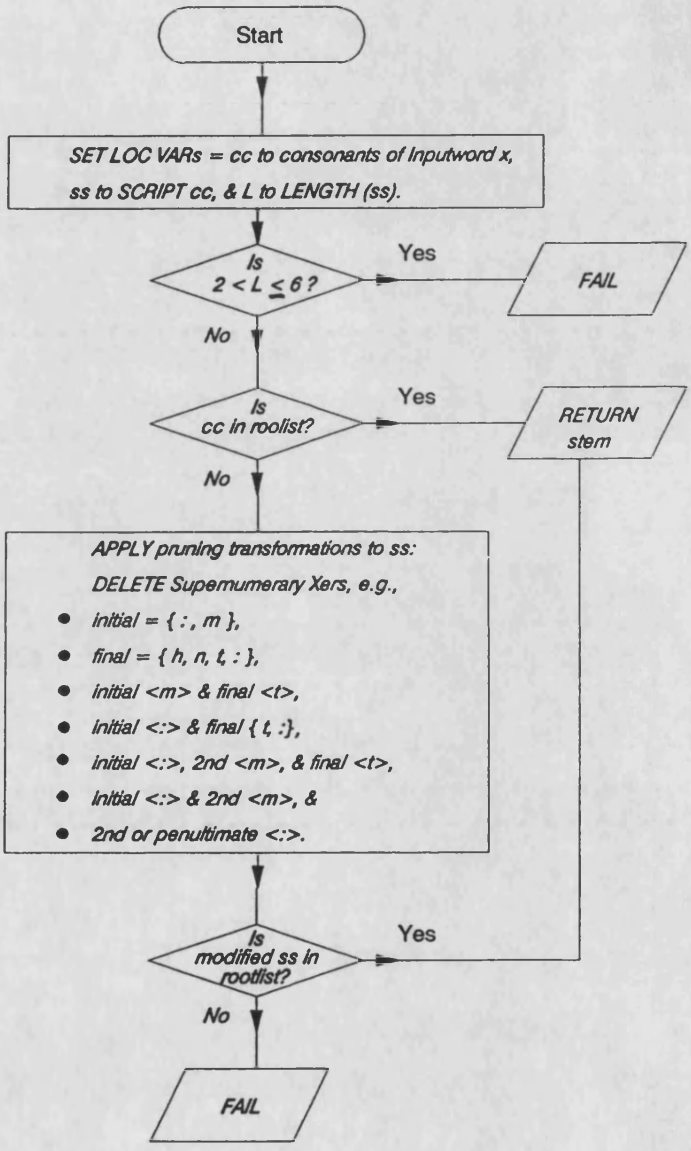


FIGURE 20: A Representation of the Stem EXTRACTion Subroutine

DERIVEQ and DERIVEZ as *DERIVEB*. DERIVEA and DERIVEB are summarized as a generalized DERIVation algorithm: NDERIVE, in Figure 21 below.

I.2.3 The Main N-Recognition Procedures

I.2.3.1 The Recognition of the Masculine Singular Category

From the description of the Nominal in Chapter 6 we can draw a formal specification for Simple NFs such as in Table 51 which follows Figure 21.

Using the initial closed subroutines and the DERIVation subroutines, we can now define procedures that are dedicated to the recognition of Simple NFs and based on the specifications of Table 51.

We start with two procedures for the recognition of masculine singular NFs: *MKMASC1*, for the Verbal CATEGORIES, and *MKMASC2*, for the non-Verbal CATEGORIES. We will refer to both as *MKMASC*. Given an inputword LL for Verbal or NN for non-Verbal, MKMASC needs to be able to recognize, if it is a masculine singular Simple NF. MKMASC recognizes such NFs without SEGMENTing them if they are Sound or Defective and have no suffixes at this point. However, MKMASC recognizes NFs by SEGMENTing them into a centre and a suffix, if they are Defective and have a Defective suffix at this point. This recognition is completed by obtaining a *valid* DERIVation for suffixless NFs, or for the centre in NFs that contain suffixes. We understand by a *valid* DERIVation here, the recognition of an actual lexical root/stem, the MATCHing of a pattern ASSOCIATed with the root/stem and the NF or centre, and the identification of the suffix, if any, among the Defective V-Suffixes: {i+, iy}.

During the parsing progress, MKMASC has to PASSON to LL the default CATEGORY and VOICE values of the LPL, the TRANSITIVITY value of the root and the TENSE value: *imperfect*. It has to PASSON to LL/NN the NG values: (M) S. In addition, a Defective LL/NN which contains a suffix will receive values for these properties: NGC, FLEXION, TYPE, HUMANITY, and PLURAL CLASS. Further, ambiguity flags will be raised in the case of the suffix 'iy'. A suffixless Defective LL/NN which is recognized here will be used later as partial input by the procedure: *MPLURALIZE*. Thus there is a contextual discrimination between default inherited values and new dynamic values in property ASSIGNment. The function PASSON was defined in Chapter 5, § I.2.3.1. The result of MKMASC is a list of: the Nominal LL/NN, the LPL, the root/stem, and the pattern.

DERIVation is free from the GPLs: "MLIST+" and "MLIST" but constrained from the exceptional GPLs: *"XLIST1"* and *"XLISTK"*, which contain Defective patterns, in that the

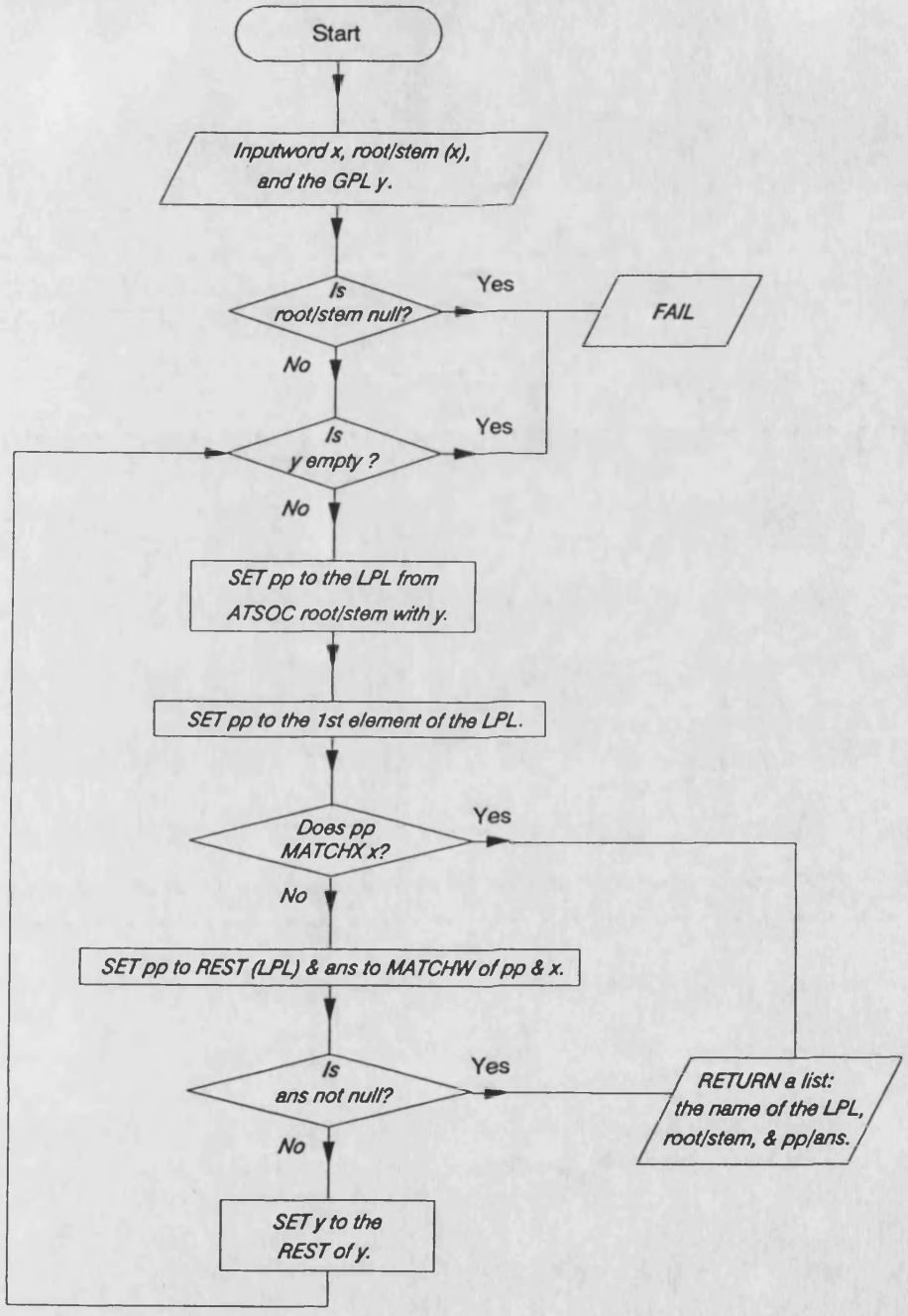


FIGURE 21: A Representation of the Nominal DERIVation Subroutine

DESCRIPTION TYPE	an abstract composite template structure.
STRUCTURE	*BASIC STRUCTURE:* a sequence of an obligatory P/R or P/S combination slot, an optional GENDER suffix (G) slot, and an obligatory NGC suffix (K) slot. *CONCATENATION ORDER:* given as P/R or P/S + (G) + K. *NF TYPES:* (a) *Sound.* (b) *Defective.* *EXAMPLES:* (a) macocac(G)K. (b) ca|ciy(G)K.
CONSTRAINTS	*GRAPHOTACTIC CONDITIONS:* include boundary conditions such as Affix Selection Rules. *NF LENGTH:* unknown but > 4. *P/R or P/S LENGTH:* the specified LENGTH. *G-SUFFIX LENGTH:* L = [2, 3]. *K-SUFFIX LENGTH:* L = [1, 5]. *P/R or P/S BOUNDARIES:* the defined boundaries. *G-SUFFIX BOUNDARIES:* >1 (G) = <t>. *K-SUFFIX BOUNDARIES:* >1 (K) = {a/i/o/u/w/y/|/a+/u+/i+}. *NF BOUNDARIES:* <1 (NF) = unknown, >1 (NF) = >1 (K). *MRP NF BOUNDARIES:* >1 (NF) = >1 (K) = {a/o/w/y}. *DUAL NF BOUNDARIES:* >1 (NF) = >1 (K) = {i/o/w/y}. *LEXICAL SPECIFICATIONS:* (a) GCs (K) = GCs (P/R or P/S); (b) ASRs (G) = ASRs (K); and (c) ASRs (K) = ASRs (P/R or P/S).
ROLE	generate NUMBER/GENDER categories of Arabic Nominals using P/Rs or P/Ss and suffixes.
SIDE EFFECTS	possible NGC disambiguation.

TABLE 51: A Formal Specification for a Simple NF

distinction of Defective Nominals at this stage will exclude them later from affixation to non-valid suffixes. From all lists, DERIVation for MKMASC is made by invoking DERIVEP/C which both retrieve consecutive first patterns in association lists inside the LPL.

The context-sensitive DERIVation from these lists also implies context-sensitive ASSIGNments of the property values ASSOCIATed with {i+, iy} if they are present.

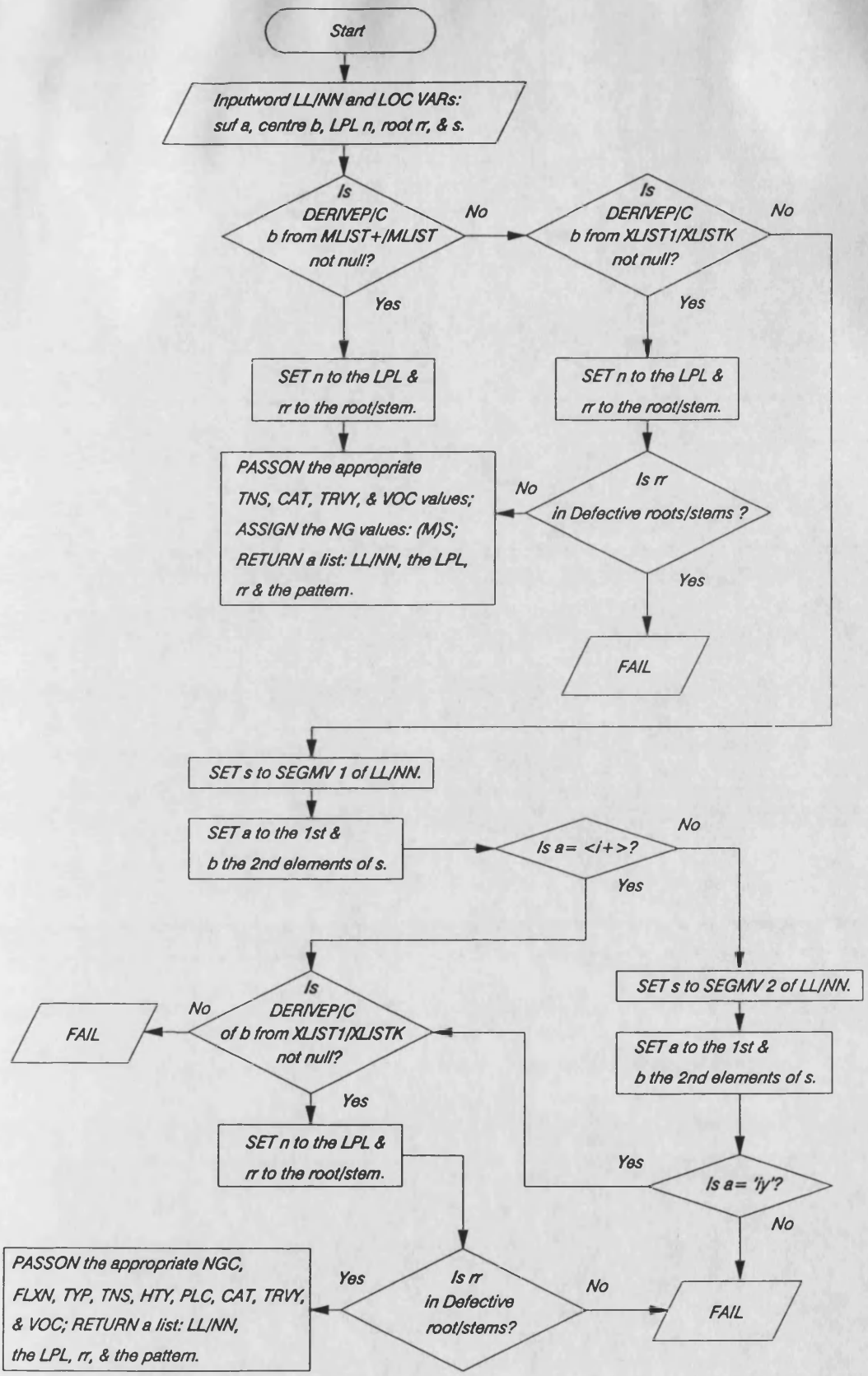


FIGURE 22:
A Representation of the Masculine Singular Recognition Procedure

SEGMENTation in MKMASC for Defective suffix identification is carried out, from smaller to larger suffixes, for the LENGTH interval: [1, 2], and from right-to-left. DERIVation then follows suffix identification. The overall MKMASC operation is a non-iterative context-sensitive procedure which is represented in Figure 22 above.

I.2.3.2 The Recognition of the Feminine Singular Category

The next procedure is called: *FEMINIZE*, which is generalized from two routines: *FEMINIZE1*—for the Verbal CATEGORIES—and *FEMINIZE2*, for other Nominals. Given an inputword LL for Verbal or NN for non-Verbal, FEMINIZE needs to be able to recognize if it is a feminine singular Simple NF which has no CASE inflection. This recognition is carried out by first attempting to obtain a valid DERIVation for the whole NF. If this attempt FAILS then the NF is SEGMENTed into a centre and a suffix, and the DERIVation is attempted for the centre after identifying the suffix among the GENDER suffixes: {at, ot}. Upon success in the last two operations, FEMINIZE has to PASSON to LL the default CATEGORY and VOICE of the LPL, the TRANSITIVITY of the root and the TENSE value: *imperfect*. It also has to PASSON to LL/NN the NG values: (F) S, and RETURN a result which is: the NF, the LPL, the root/stem, and the pattern.

DERIVation is attempted first from the exceptional GPLs: *"XLIST3"*, *"XLISTA"*, *"XLISTC"*, *"XLISTE"*, and "XLISTK" which hold LPLs with association lists for stem groups that have constrained DERIVations such as Abbreviated Nominals and stems that may have non-morphologically derived feminine NFs. If this DERIVation is not successful, then simple transformations are applied to Defective patterns in order to fit a MATCH at database level and the DERIVation is attempted again. For instance, the NF 'bina|yat', "building" has the pattern 'cica|y', but the MF-Pattern stored for 'bn', "building" in the database is 'cica|:' and <:> has to be substituted with <y> in order to get the MATCH, hence the context-sensitivity of FEMINIZE.

If after modification, there is still no MATCH then DERIVation is attempted from the regular GPLs: "!*FLIST" or "FLIST". All DERIVations invoke DERIVEP or DERIVEC both of which retrieve consecutive first patterns in association lists inside LPLs.

The process of DERIVation and SEGMENTation in FEMINIZE is carried out in right-to-left order: DERIVation of NF, identification of suffix, DERIVation of centre. FEMINIZE is a non-iterative context-sensitive procedure which is represented in Figure 23 below.

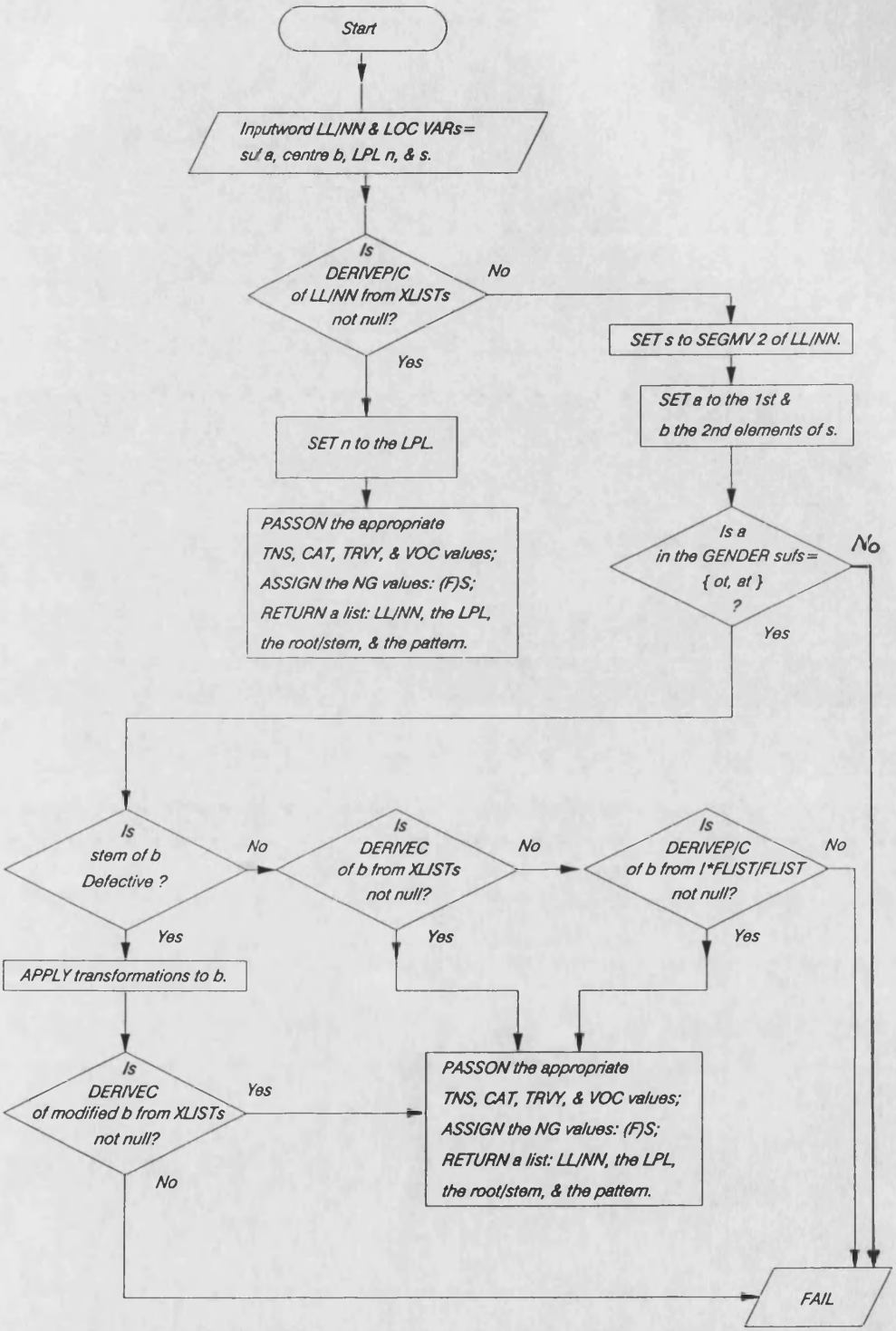


FIGURE 23:
A Representation of the Feminine Singular Recognition Procedure

I.2.3.3 The Recognition of the Dual Category

DUALIZE is a unified procedure for the recognition of the dual masculine and feminine categories and which consists of *DUALIZE1*, for the Verbal CATEGORIES, and *DUALIZE2*, for other Nominals. Given an inputword LL for Verbal or NN for non-Verbal, DUALIZE has to recognize if it is a dual Simple NF by SEGMENTing the NF into a centre and a suffix. After identifying the suffix among the set of dual suffixes, DUALIZE attempts to obtain a valid DERIVation for the centre from a restricted set of exceptional GPLs: *"XLISTB"*, *"XLISTJ"*, "XLISTK", *"XLISTL"*, and *"XMLIST"* holding LPLs for stem groups with constrained DERIVations such as dual-only and Defective Nominals.

If the attempted DERIVation is not successful then MKMASC1 is invoked and applied to the centre, and if the result is not successful then FEMINIZE1 is invoked. All DERIVations invoke DERIVEP/C which use consecutive first patterns in LPLs. If all DERIVations from exceptional GPLs FAIL, even after transformations on the centre, then FEMINIZE2 is called. If FEMINIZE2 FAILS, then the suffix-LENGTH interval is incremented by 1 and the whole process is resumed. Thus, DUALIZE performs an iteration for the LENGTH interval: [2, 5] of the dual suffix, as well as a recursion in that it invokes the MKMASC and FEMINIZE routines on parts of the input it is working on.

Upon success, DUALIZE has to PASSON to LL the inherited CATEGORY, TENSE, TRANSITIVITY, and VOICE values of the centre, ASSIGN to LL/NN the NUMBER, CASE, and FLEXION of the suffix, the GENDER of the centre, ASSIGN to NN the CATEGORY of the LPL, and RETURN a list: the NF, the LPL, the root/stem, and the pattern.

The process of SEGMENTation and DERIVation in DUALIZE is carried out in a right-to-left order: identification of suffix, DERIVation of centre, and recursive calls to MKMASC first and then to FEMINIZE. Thus, DUALIZE is both an iterative and a recursive procedure which is outlined in Figure 24 below.

I.2.3.4 The Recognition of the Masculine Regular Plural Category

MPLURALIZE is a unified procedure for the recognition of the Masculine Regular Plural category and which consists of *MPLURALIZE1*, for Verbals, and *MPLURALIZE2*, for other Nominals. Given an inputword LL for Verbal or NN for Nominal, MPLURALIZE has to recognize if it is a MRP Simple NF by SEGMENTing it into a centre and a suffix. If the suffix is identified among the set of Defective plural suffixes, MPLURALIZE has to obtain a valid DERIVation for the centre from a restricted set of GPLs: *"XLIST2"*, "XLISTC", *"XLISTI"*, "XLISTJ", and "XLISTK", holding LPLs for stem groups with restricted pluralization, such

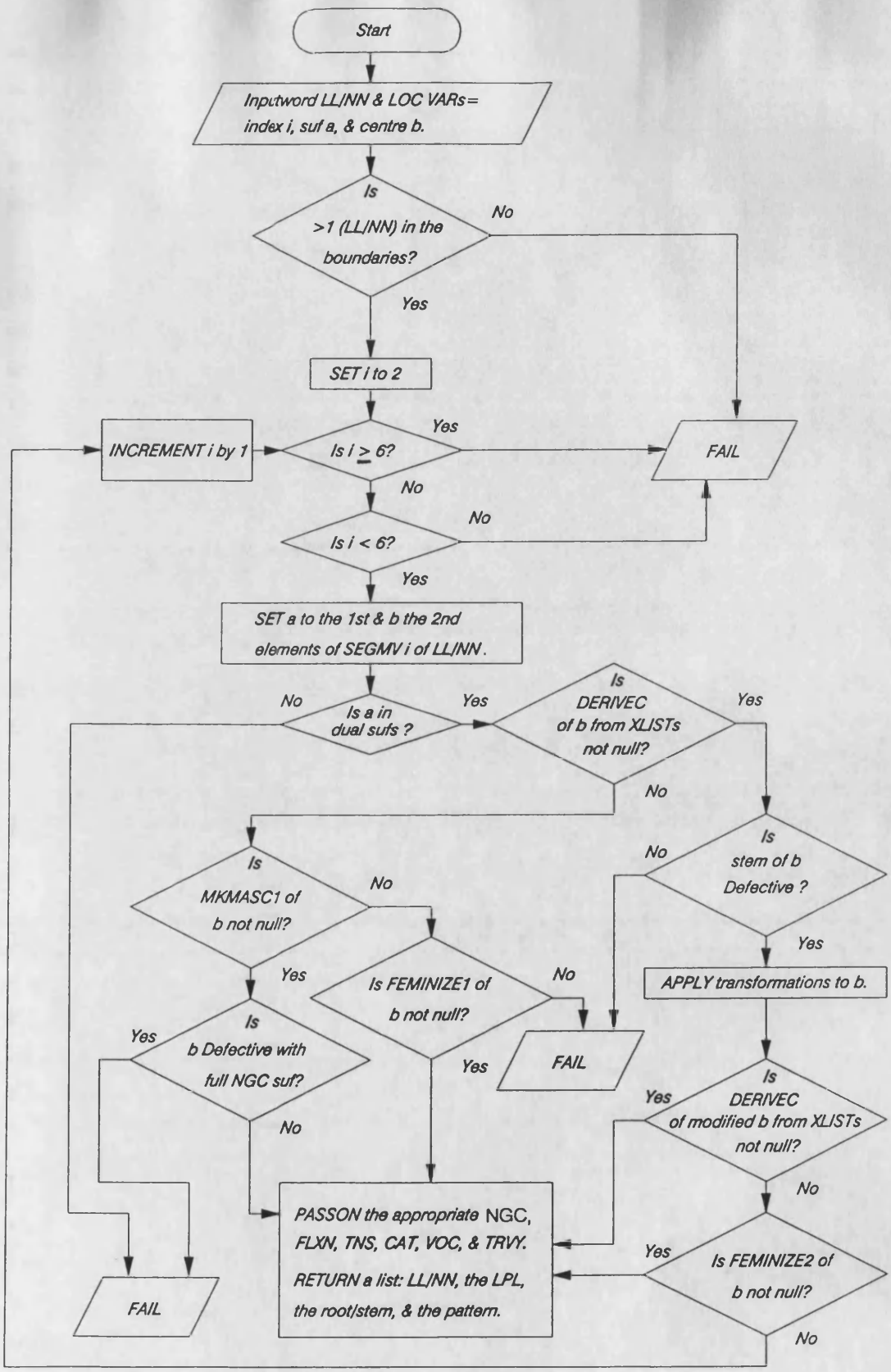


FIGURE 24: A Representation of the Dual Recognition Procedure

as Defective and non-human stems. If this FAILS, then simple transformations are applied at the NF boundaries to accommodate ASRs and other boundary conditions (as described in Ch. 6). DERIVation is then attempted for the modified centre. If this still FAILS, then an attempt to identify the suffix among the MRP suffixes is made, and the whole DERIVation process is resumed.

All DERIVations invoke DERIVEP/C which use consecutive first patterns of LPLs. If all DERIVations FAIL, then the suffix LENGTH interval is incremented by 1 and the whole operation restarted. Thus, MPLURALIZE is an iteration for the LENGTH interval: [1, 5] of the plural suffix. It makes recursive calls to the MKMASC routine on parts of its input. Upon success, MPLURALIZE has to PASSON to LL the inherited or contextual CATEGORY, VOICE, TENSE, and TRANSITIVITY values, ASSIGN to LL/NN the CASE, FLEXION, PLURAL CLASS, and HUMANITY of the suffix, then ASSIGN to NN the CATEGORY of the LPL, ASSIGN to LL/NN the NG: (M) P, and RETURN a list: the NF, the LPL, the root/stem, and the pattern.

The process of SEGMENTation and DERIVation in MPLURALIZE is a right-to-left one: suffix identification, centre DERIVation, recursion on MKMASC. Like DUALIZE, MPLURALIZE mixes direct DERIVation with recursive calls to other procedures, e.g., MKMASC. Both types of DERIVational and recursive operations occur within the iteration for a given suffix-LENGTH interval. MPLURALIZE is illustrated in Figure 25 below.

I.2.3.5 The Recognition of the Feminine Regular Plural Category

The next procedure is unified *FPLURALIZE* consisting of *FPLURALIZE1*, for Verbals, and *FPLURALIZE2*, for other Nominals. Given an inputword LL for Verbal or NN for Nominal, FPLURALIZE has to recognize if it is a Feminine Regular Plural Simple NF without CASE inflection by SEGMENTing the NF into a centre and a suffix. After identifying the suffix as the FRP GENDER Suffix: 'a|t', FPLURALIZE attempts to obtain a DERIVation for the centre.

DERIVation is attempted, first, by invoking DERIVEQ, and hence using consecutive tails of LPLs which are found in exceptional GPLs for Nominals with DERIVation from BP patterns; then, by invoking DERIVEP/C, and hence using consecutive first patterns in LPLs which hold stems with constrained DERIVation, such as Defective Nominals with compulsory simple boundary transformations, and which are found in exceptional GPLs: "XLISTA", "XLISTC", and *"XLISTD"*. For example, the NF 'bina|ya|t', "buildings" has the pattern 'cica|y', but the MF-Pattern stored for 'bn', "building" in the database is 'cica|:', and <:> has to be substituted with <y> in order to get the MATCH.

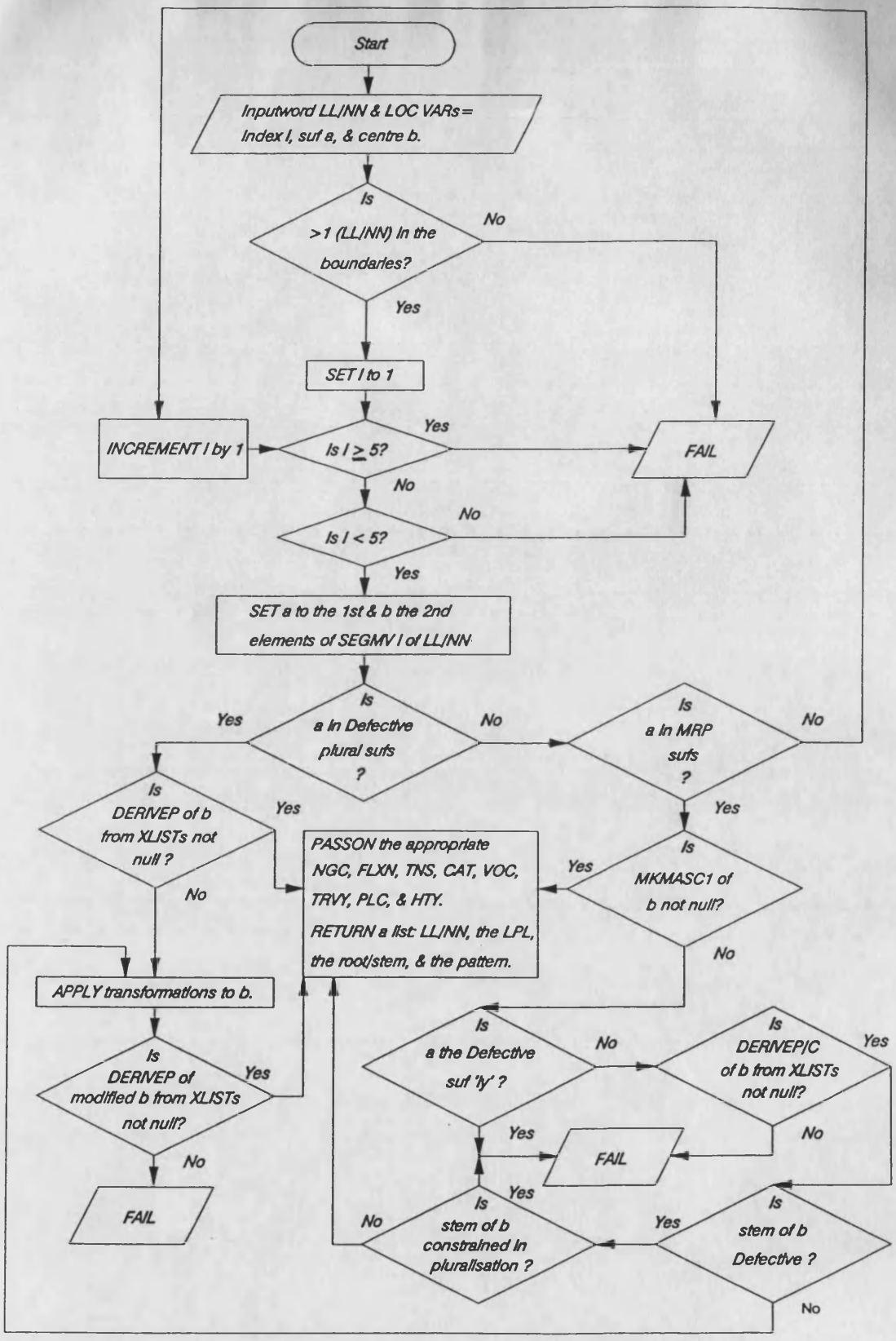


FIGURE 25:
A Representation of the Masculine Regular Plural Recognition Procedure

Simple transformations are then applied and if DERIVation FAILS for the modified centre, then it is attempted from the GPLs: "!*FLIST" and "FLIST". Whenever a DERIVation is successful, stems are divided into two groups: human, to which a HUMANITY value is ASSIGNed, and non-human. For both groups, FPLURALIZE has to ASSIGN to LL/NN the NG values: (F) P, the PLURAL CLASS value: *rfp*. It has to ASSIGN to LL the CATEGORY and VOICE of the LPL, the TRANSITIVITY of the root, and the TENSE: *imp*. It has to ASSIGN to NN the CATEGORY of the LPL and then RETURN a list of: the NF, the LPL, the root/stem, and the pattern. If all DERIVations FAIL, then FPLURALIZE FAILS.

The process of SEGMENTation and DERIVation is attempted only once since the FRP suffix has a constant LENGTH: [3] and the operation of FPLURALIZE is a context-sensitive non-recursive and non-iterative right-to-left procedure which is represented in Figure 26 below.

I.2.3.6 The Recognition of the Broken Plural Category

The next procedure is *BPLURALIZE* which is for all Nominals. Given an inputword NN for Nominal, MPLURALIZE has to recognize if it is a masculine, feminine, or neutral Simple NF. This recognition is achieved without SEGMENTation for Sound and Defective NFs that have no suffixes at this point. Otherwise, BPLURALIZE proceeds by SEGMENTing the NF into a centre and a suffix for other Defective NFs.

DERIVation is attempted directly for the whole NF, just in case it is suffixless, or for the centre just in case it contains a suffix. After identifying the suffix in the Defective suffixes: {i+, iy}, DERIVation is started from the GPLs: *"MDLIST"* and *"FDLIST"*; and then from the constrained GPLs: "XLISTA", "XLISTB", "XLISTC", *"XLISTF"*, *"XLISTG"*, *"XLISTH"*, "XLISTI", and "XLISTK", always invoking the lower routine DERIVEZ which retrieves patterns from consecutive tails of LPLs.

There are no transformations in BPLURALIZE, but this is also a context-sensitive procedure in that distinction is made between human and non-human stems for property ASSIGNment. Other contextual ASSIGNments here include CASE, FLEXION, TYPE, and CATEGORY for Defective NFs. Inherited and contextual values are also ASSIGNed to all NFs for NUMBER, GENDER, and CATEGORY. This latter ASSIGNment is done using the suffix, if any, and the LPLs. Further, all ASSIGNments are made upon success in DERIVation and are followed by RETURNing the result which is a list including NN, the LPL, the stem, and the pattern.

SEGMENTation in BPLURALIZE is similar to that in MKMASC, in that Defective suffix identification is carried out from smaller to larger suffixes for the LENGTH interval: [1, 2] in a right-to-left order. BPLURALIZE is represented in Figure 27 below.

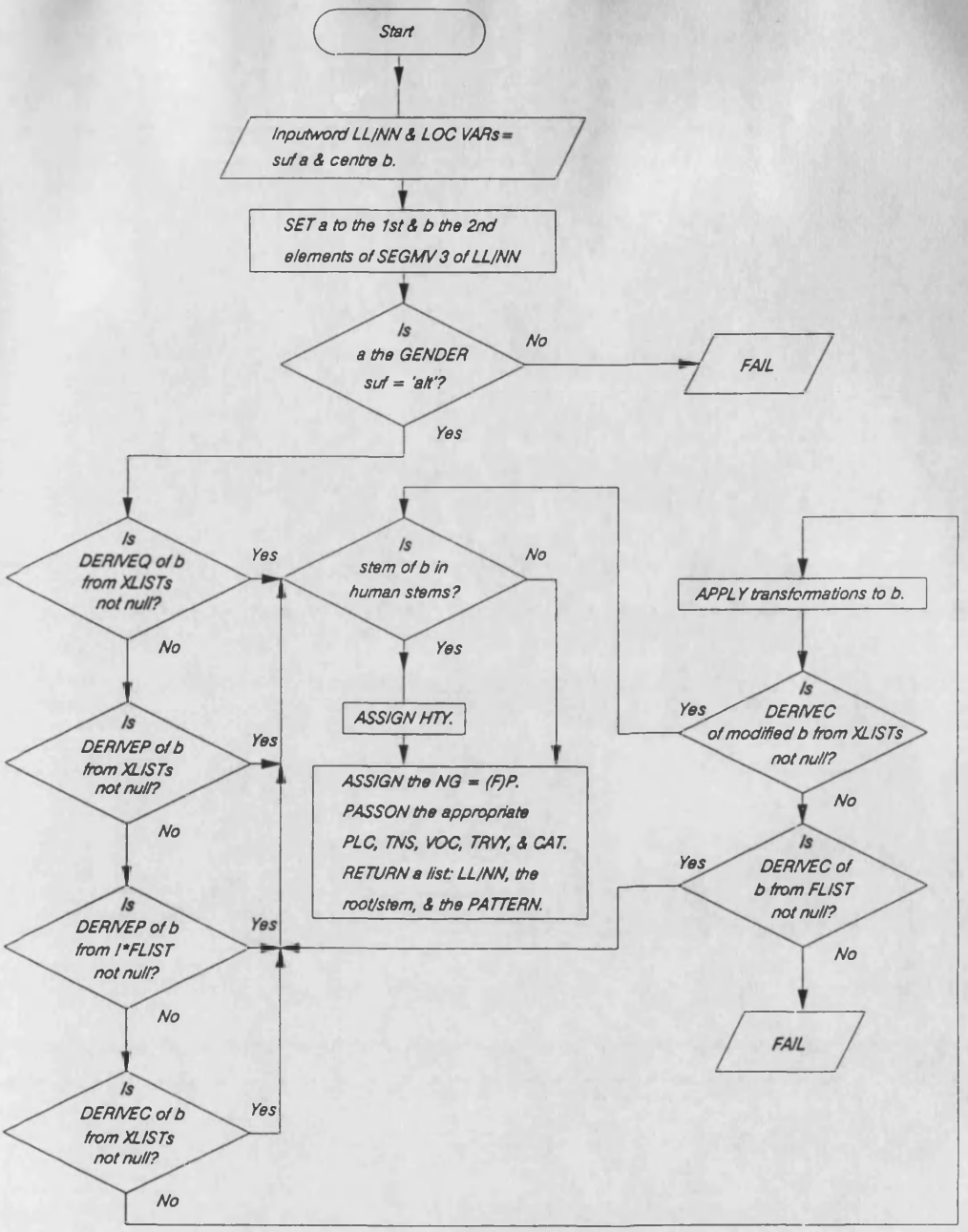


FIGURE 26:
A Representation of the Feminine Regular Plural Recognition Procedure

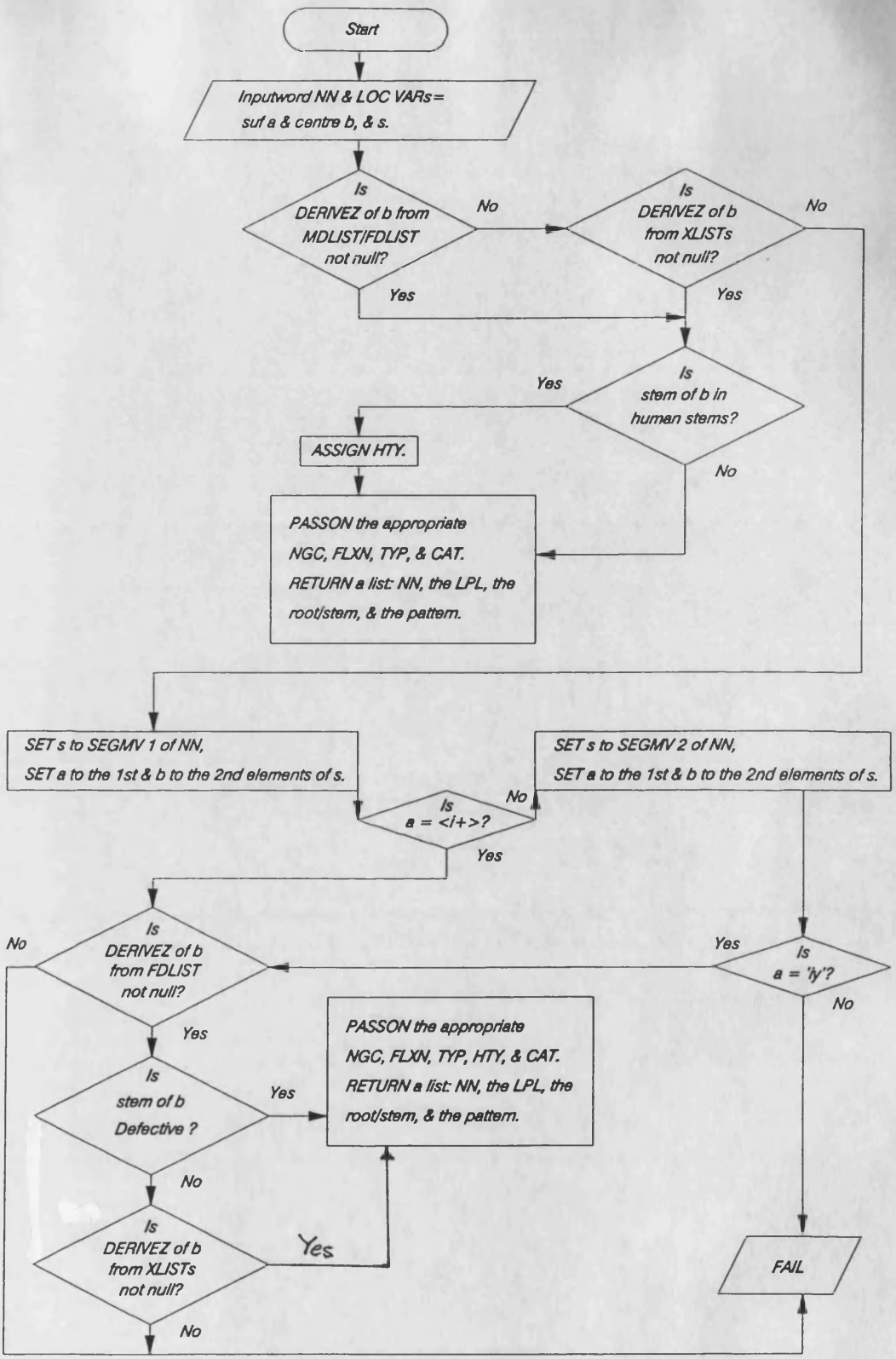


FIGURE 27: A Representation of the Broken Plural Recognition Procedure

I.2.3.7 NUMERICAL, TRANSITIVITY, and HUMANITY Values MODIFICation

Having defined a number of procedures for the recognition of NUMBER-GENDER categories in Simple NFs, we can now envisage *traffic control* functions such as *NPARSE* and *LPARSE*. NPARSE will RETURN the first non-NIL result RETURNed by MKMASC2, FEMINIZE2, BPLURALIZE, FPLURALIZE2, MPLURALIZE2 or DUALIZE2. If NPARSE FAILS, then LPARSE will RETURN the first non-NIL result RETURNed by MKMASC1, FEMINIZE1, FPLURALIZE1, MPLURALIZE1 or DUALIZE1.

In Table 46, we specified NUMBER contexts for which different NUMERICAL VALUES are calculated for each Numeral. In order to avoid complications, the NPARSE procedure does not take account of these contexts. The task given to the context-sensitive declarative procedure *MVACT*, which is a NUMERICAL VALUE calculator which invokes NPARSE, is to discriminate between these contexts and ASSIGN the implied NUMERICAL VALUE to Numeral Nominals only.

MVACT also takes this opportunity to ASSIGN HUMANITY values to singular Nouns, in which we included non-Derived Substantives and Tlocatives. We will discuss, in Chapter 12, the distinction of HUMANITY values and its importance for Nominal-Numeral agreement. Similarly, in Chapter 6, § II.2.5, we specified homonymy contexts for TRANSITIVITY ASSIGNment which were not taken into account by the LPARSE procedure. The task given to the context-sensitive declarative procedure TVACT2, which stands for TRANSITIVITY *act*ion and which invokes LPARSE, is to take account of these contexts and ASSIGN the appropriate TRANSITIVITY value to Verbals. TVACT2 also implements a DEMOTion of TRANSITIVITY values in a parallel way to TVACT1 under passivization. Further, it ASSIGNS HUMANITY values to the singular Verbal in a parallel way to MVACT.

I.2.3.8 CASE-Inflection Stripping

The *FLEX* routine is a context-sensitive declarative procedure which has the task of stripping off CASE-only inflection suffixes in Simple NFs. Thus, FLEX acts on the output of MKMASC, FEMINIZE, FPLURALIZE, and BPLURALIZE; but it does not act on the output of MPLURALIZE or DUALIZE, since the output of these latter two is NFs with NGC suffixes, i.e., NFs that are already inflected for CASE. We have already seen that the output of MKMASC, FEMINIZE, and BPLURALIZE can be (M, F) BP Sound or Defective NFs, whereas the output of FPLURALIZE is FRP NFs.

FLEX imposes constraints on the type of CASE inflection carried by Diptotes of Type 1, Diptotes of Type 2 such as FRPs, and Defectives ending in 'ay' and 'iy'. Further, Defectives

with 'iy' are ASSIGNed homonymic values for NUMBER, HUMANITY, DEFINITENESS, REFERENCE, and CATEGORY (as specified in Ch. 6, § II.2.3).

Another type of NF recognized by FLEX is the Proper Noun carrying a CASE inflection but not ending in 'ay', since Proper Nouns are not recognized by any other procedures described so far. The recognition tasks performed by FLEX are carried out, in a right-to-left manner, by stripping off the CASE suffix first, then invoking MVACT or TVACT2 on the centre. However, with Abbreviated and non-accusative Weak NFs (both of which already carry Defective CASE suffixes by this stage), MVACT or TVACT2 is invoked on the whole NF, but a suffix of LENGTH: [2] is inspected to determine the structural context for property ASSIGNments.

The values ASSIGNed or PASSed ON here are for these properties: NGC, FLEXION, HUMANITY, DEFINITENESS, REFERENCE, CATEGORY, VALUE, and TYPE. In addition, TENSE, TRANSITIVITY, and VOICE values are ASSIGNed to Verbals. The flag value *rnc* (i.e., restricted to nominative and accusative) is ASSIGNed to Diptotes of Type 1 in order to inhibit oblique marks on such Nominals. Note, however, that the oblique interpretation, viz., as deep CASE value, is still allowed. The result RETURNed by FLEX is a list of the inputword. FLEX is illustrated in Figure 28 below.

I.2.3.9 Determiner Stripping and the Implementation of the Affixation Conditions

From the descriptions in Chapter 6, § II.1.6, and § II.1.7, we imply a formal specification for Complex NFs which are DNFs, ANFs, and PNFs. Table 52 below spells out this specification.

To satisfy the requirements of Table 52, we need further procedures to use the routines outlined so far in order to recognize Complex NFs.

We start with *DEFINITIZE1*, which is a context-sensitive iterative procedure geared for the recognition of Simple NFs with Determiners. However, it starts with the recognition of SPs, DPs, and Simple NFs devoid of Determiners. All these structures have to satisfy initial and final boundary conditions as in Table 52. DEFINITIZE1 also implements the constraints on D$NF affixation referred to as *ASRs* and including graphemic assimilation.

Further, DEFINITIZE1 disambiguates property values for NGC, FLEXION, DEFINITENESS, and CATEGORY as far as Defective NFs are concerned. The context for such disambiguation is provided by the conditions of affixation of Determiners to NFs (as described in Ch. 6, § II.2.4). The identification of structural contexts allows DEFINITIZE1 also to ASSIGN other property values for TYPE, HUMANITY, and NUMERICAL VALUE. The result RETURNed by DEFINITIZE1 is the result of FLEX, if there is no Determiner, and a list of the NF otherwise.

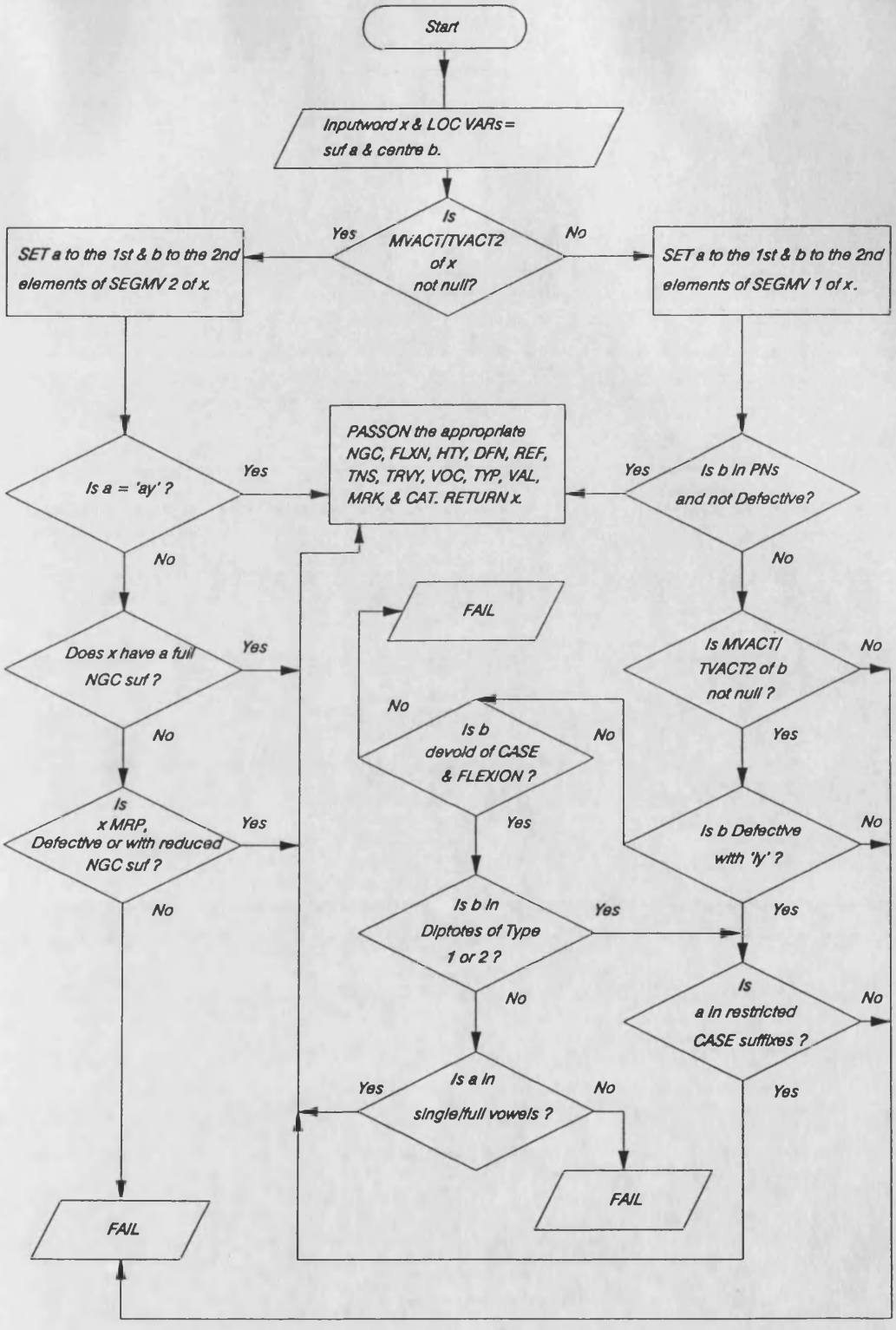


FIGURE 28: A Representation of the CASE Inflection Procedure

DESCRIPTION TYPE	an abstract composite template structure.
STRUCTURE	*BASIC STRUCTURE:* a sequence of optional affix slots: Pr, D, and GP; and an obligatory NF slot. *CONCATENATION ORDER:* given as (Pr) + (D) + NF, or (Pr) + NF + (GP). *CNF TYPES:* (a) *Indefinite.* (b) *Definite.* (c) *Annexed.* (d) *Prepositional.* *EXAMPLES:* (a) i. Modifier NF. (a) ii. Explicitly Indefinite NF. (b) Definite NF. (c) Annexed NF. (d) i. P$Modifier NF. (d) ii. P$Explicitly INF. (d) iii. P$Definite NF. (d) iv. P$Annexed NF.
CONSTRAINTS	*GRAPHOTACTIC CONDITIONS:* include boundary conditions such as Vocalic Compatibility and Affix Selection Rules. Other conditions include CASE GOVERNMENT. *CNF LENGTH:* unknown but > 4. *NF LENGTH:* the specified LENGTH. *Pr LENGTH:* L = [2]. *D LENGTH:* L = [1, 4]. *GP LENGTH:* L = [2, 5]. *CNF BOUNDARIES:* *if* Pr then <1 (CNF) = <1 (Pr), else *if* D then <1 (CNF) = <1 (D); and *if* GP then >1 (CNF) = >1 (GP), else >1 (CNF) = >1 (NF). *NF BOUNDARIES:* the defined boundaries. *Pr BOUNDARIES:* <1 (Pr) = {l/b/k}. *D BOUNDARIES:* <1 (D) = {:/l}. *GP BOUNDARIES:* >1 (GP) = {a/i/o/u/y/l}. *LEXICAL SPECIFICATIONS:* (a) *if* Pr then, CASE (Pr) = CASE (CNF); and (b) *if* Pr & D then, VCo (Pr) = VCo (D); and (c) *if* D then, ASRs (D) = ASRs (NF); and (d) *if* GP then, VCo (GP) = VCo (NF) & GCs (GP) = GCs (NF).

TABLE 52: A Formal Specification for a Complex NF

CONSTRAINTS	*ADDITIONAL CONDITIONS:* (a) *if* D then GP $\longrightarrow$ Ø; and (b) *if* GP then D $\longrightarrow$ Ø.
ROLE	provide a template for the realization of NFs with or without attached Pr, D, or GP.
SIDE EFFECTS	possible NGC, FLXN, DFN, PLC, HTY, REF, and CAT disambiguation.

TABLE 52: A Formal Specification for a Complex NF (Contd)

DEFINITIZE1, as opposed to other N-Procedures so far, proceeds in a left-to-right manner stripping off a Determiner of a LENGTH interval: [1, 4] and invoking FLEX on the centre until a result is obtained. The index *i* is SET to 4 as an upperbound and is decreased by 1 each time there is a FAILure until the threshold 1 is reached. At each iteration, a prefix is identified among the allomorphs of the Determiner and checked for Lunar or Solar compatibility depending on the initial character of the centre and in accordance with the conditions in Chapter 6, § II.1.6.1. In addition, the centre is FLEXed and subject to the constraint: *D$NF where NF [+def]. Figure 29 below represents DEFINITIZE1.

I.2.3.10 Genitive-Pronoun Stripping and the Implementation of the Affixation Conditions

The second procedure whose task it is to recognize Complex NFs in accordance with the specifications of Table 52 is *DEFINITIZE2*. This is a context-sensitive iterative procedure that has to recognize Complex NFs attached with optional GPs and implement constraints on NF$GP referred to as ASRs. If a CNF satisfies the entry condition of conforming to the final boundary requirements, DEFINITIZE2 proceeds in a right-to-left manner to strip off GPs of a LENGTH interval: [2, 5] starting at the threshold and incrementing the index by 1 each time an iteration FAILS. This FAILure happens if the stripped suffix is not in the set of GPs, or if the centre does not satisfy parsing conditions at lower procedures.

With the first-PERSON-singular GP (with which the NF does not admit CASE inflection suffixes), the lower procedures invoked on the centre are MVACT, TVACT2, and FLEX. If a result is not obtained from these calls, then simple boundary transformations are applied in accordance with Chapter 6, § II.1.6.2, and the said procedures are invoked again on the modified centre. With the rest of the GPs (with which the NF does admit CASE inflection suffixes), the FLEX procedure is invoked on the centre. If the result is still a FAILure, then simple transformations are performed and FLEX is invoked again. In addition, all parsed centres have to satisfy the DEFINITENESS constraint: *NF$GP where NF [+def].

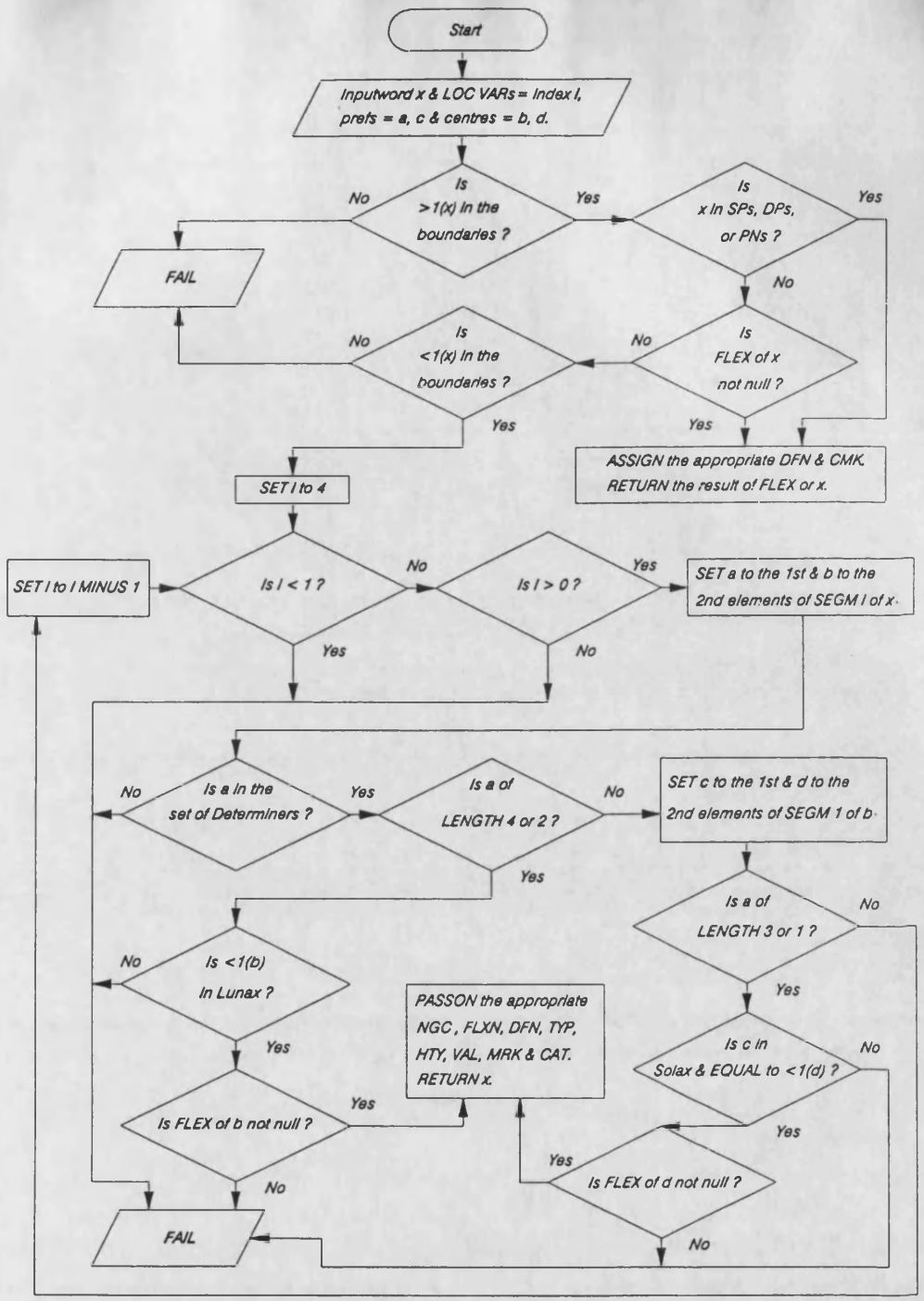


FIGURE 29: A Representation of the DNF Recognition Procedure

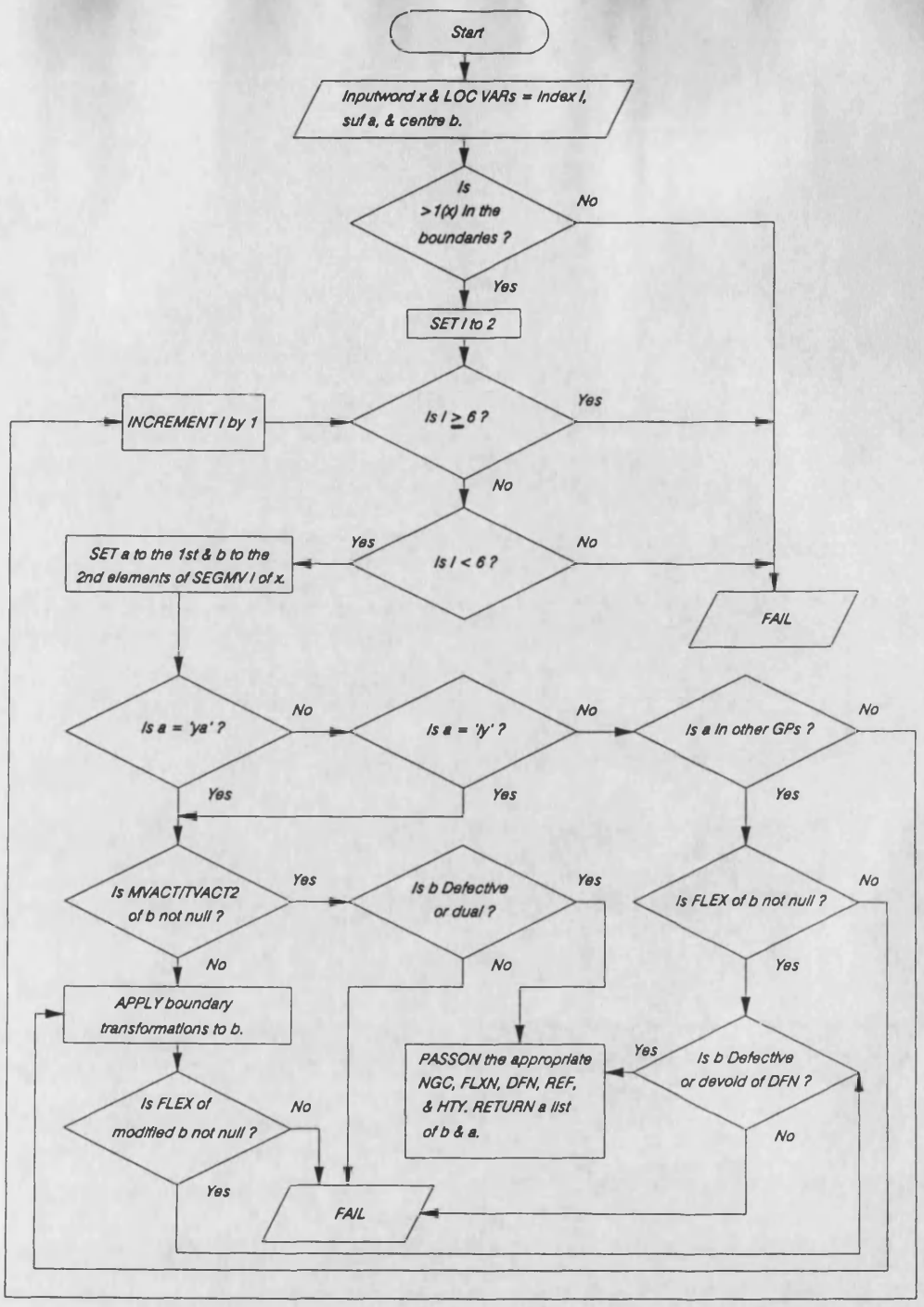


FIGURE 30: A Representation of the ANF Recognition Procedure

Upon satisfying all these requirements, DEFINITIZE2 ASSIGNS property values for NGC, FLEXION, DEFINITENESS, REFERENCE, and HUMANITY. It also performs some disambiguation for these properties just in case the centre is Defective and in accordance with Chapter 6, § II.2.4. The result RETURNed by DEFINITIZE2 is a list of the centre and the suffix. DEFINITIZE2 is outlined in Figure 30 above.

I.2.3.11 Graphotactic and TRANSITIVITY Filtering and Categorial Disambiguation for Complex NFs

The Vocalic Compatibility (VCo) and TRANSITIVITY conditions (in Ch. 6, § II.1.6.1 and § II.1.6.2) have not been implemented so far. FILTER4 is a declarative context-sensitive scheme which enforces the observation of these conditions in the output of DEFINITIZE2. Similarly to FILTER3, *FILTER4* distinguishes collocutive and exlocutive ANFs, but also excludes all intransitive NFs from annexation with GPs. Exlocutive ANFs are then forced to comply with VCo for each allomorphic variant of a given GP.

Besides the above type of filtering, FILTER4 also performs some categorial disambiguation on the output of DEFINITIZE1 and 2 taking into account the type of FLEXION carried by the CNF and the FLEXION-CATEGORY correlation as described in Chapter 6, § I.1.3.5. The FAILure of FILTER4 is indicated by the RETURN of an error message which explains the nature of the eventual violation in the input, while success is indicated by the RETURN of the inspected and modified input of DEFINITIZE1 or 2.

I.2.3.12 Genitivization and CASE Disambiguation

The third procedure to recognize Complex NFs in accordance with the specifications of Table 52 is *GENITIVIZE2*. This is a procedural context-sensitive scheme that has to recognize a Complex NF which is a composite structure of an optional Bound Preposition followed by a DNF or an ANF. GENITIVIZE2 also has to implement CASE constraints on the affixation of Prepositions to CNFs and VCo conditions which include elision as well as conditions governing the juxtaposition of Prepositions to Determiners just in case the Preposition in question is 'li ', "to, for, ...".

After attempting FILTER4 for the whole inputword and if a PNF satisfies the entry condition of conforming to the initial boundary requirements, GENITIVIZE2 proceeds in a left-to-right direction. First, it attempts the identification of a prefix of constant LENGTH: [2] in the set of Bound Prepositions. Then, it invokes FILTER4 on the centre.

If these latter operations succeed then the CNF is inspected for observation of the CASE requirements and the Graphotactic Conditions: DNFs following P have to be *bound*, i.e, observe

elision. Such NFs are said to be *bound* as they cannot occur without being attached with P. Other DNFs are said to be free and can attach with the Prepositions: 'bi', "in, at, ...", and 'ka', "as", or occur freely. All CNFs have to carry oblique CASE except Numerals with temporary indeclinability. All these rules are implemented in accordance with the specifications in Chapter 6, § II.1.6 and § II.1.7.

Following the identification of the structural context, GENITIVIZE2 performs a disambiguation in CASE values for NFs following Prepositions and carrying ambiguous or neutral CASE. GENITIVIZE2 also has to PASSON to the inputword x the property heritage: NG, FLEXION, NUMERICAL VALUE, DEFINITENESS, HUMANITY, TYPE, REFERENCE, and CATEGORY. Finally, it RETURNS a result which is x, for non-Annexed CNFs, and a list of the CNF and the GP, for Annexed CNFs. GENITIVIZE2 is represented in Figure 31 below.

II SYNTHESIS OF THE N-COMPONENT

II.1 THE LOGICAL STRUCTURE OF THE N-COMPONENT

II.1.1 The Structure of the N-Complex

In Appendix A, § B, we gave a list of Nominal pattern distributions for all the stems included in this study. We divided these patterns into groups and according to a N-Typology described in Chapter 6, § I. In § II of Chapter 6, we used the collapsing method to make generalized statements about the patterning and structure of Sound and Defective Verbal and non-Verbal Nominals.

The linguistic analysis of N-Complexes involved an account of their morphological structure and distinguished several kinds of affix: optional *inner* G suffixes and obligatory *inner* K suffixes inside Simple NFs; and optional *outer* affixes: Bound P, D, and GPs inside Complex NFs. For each of these affixes, we specified various conditions of allomorphic variation: Graphotactic Conditions including Affix Selection Rules and Vocalic Compatibility consisting of assimilation, elision, as well as other graphemic harmony rules. We also specified syntactic conditions including TRANSITIVITY for Verbals; and lexical specification consisting in CASE GOVERNMENT rules. Further, we noted correlation conditions between certain affixes and such processes as categorization, and structural dependencies including exclusive conditions for definitization.

For each main segment, or N-Centre, we detailed DERIVation conditions including root/stem and pattern types, regularized processes (*'qiya|siyyat'*) for the generation of

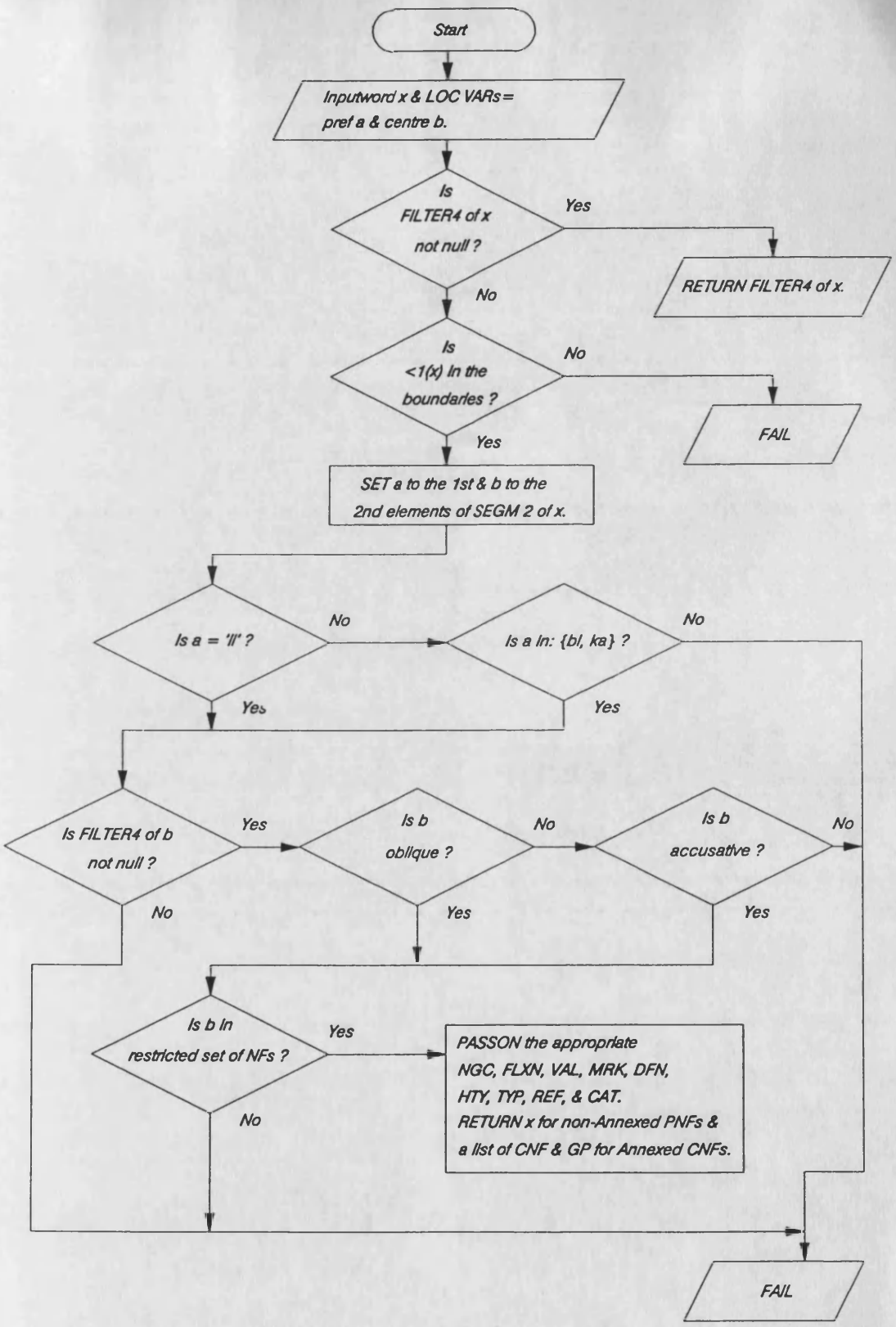


FIGURE 31: A Representation of the PNF Recognition Procedure

NUMBER, GENDER, and CASE categories and non-regularized processes such as Broken Plural patterning. For all these generalizations, we had to introduce refinements in order to account for exceptional cases of structural and property homonymy. The above analysis led us to a statement of constrained Rewrite Morphological Rules which we referred to as Annotated MS rules.

In this Chapter, we specified for each of the described Nominal morphological structures, a property heritage including NGP, CASE, FLEXION, DEFINITENESS, HUMANITY, TYPE, REFERENCE, NUMERICAL VALUE, PLURAL CLASS, and CATEGORY values, as well as VOICE, TRANSITIVITY, and TENSE values for Verbals only. These values were divided into default (or database) values, contextual (or run-time) values, neutral (or syncretic) values, and ambiguous (or flagged) values.

We then deduced from the account in Chapter 6, formal specifications for Complex NFs with definitions for CNF types; constituent structure; affixation of NFs with Determiners, GPs, and Bound Prepositions; simple boundary transformations; LENGTH constraints; and lexical specifications for TRANSITIVITY, CASE GOVERNMENT, and Graphotactic Conditions governing intra-morphemic juxtaposition. We then described Cambridge LISP facilities for the representation of these data structures and the encoding of the morphological rules and constraints for Arabic Nominals. Figure 32 below shows a general model of N-Complex in M.S.A.

II.1.2 The Structure of the N-Processor

The computational procedures of the N-Processor as described in this Chapter implement the linguistic description provided in Chapter 6. Similarly to the V-Processor, the N-Processor implements boundary specifications as entry conditions; and LENGTH specifications as numerical intervals for which iterations are performed starting at the threshold and exiting at the upperbound. Affixation conditions are also implemented as context-sensitive declarative procedural filters enforcing the observation of these constraints in the output of lower routines; and CASE GOVERNMENT conditions are implemented as direct requirements on the NF in question.

The N-Processor also handles boundary changes through the use of simple efficient transformations which insert or DELETE Supernumerary characters and affixes in order to simulate exact MATCHes with database forms such as stems and patterns. The meticulous identification of concatenation contexts involving roots/stems, patterns, and affixes allows the N-Processor to ASSIGN contextual property values; to PASSON appropriate default property values from smaller to larger structural units; and to disambiguate homonymic property values where possible, or raise ambiguity flags elsewhere.

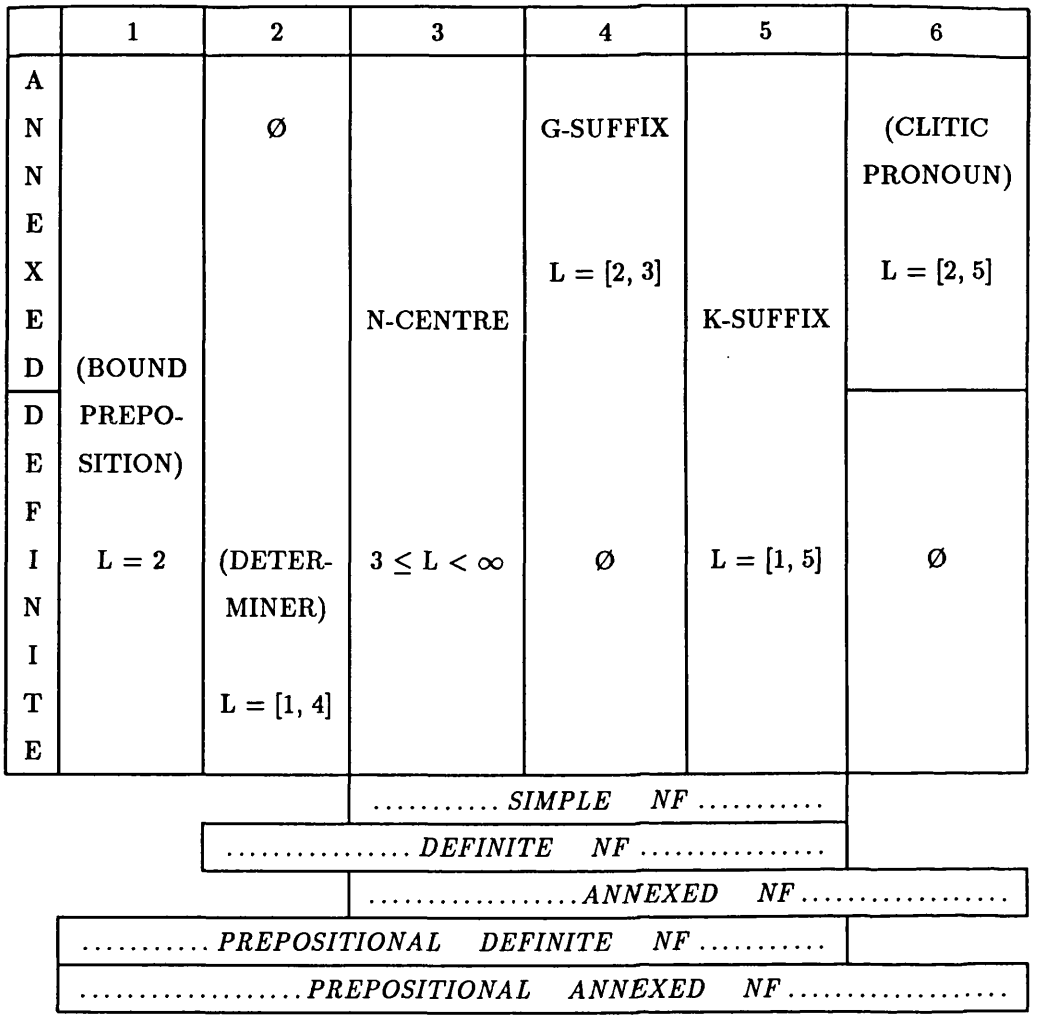


FIGURE 32: A General Model for the N-Complex in M.S.A.

The identification of concatenation and property contexts in the N-Processor proceeds according to this logical framework:

<DERIVation>,
    <case: root/stem context>,
<SEGMENTation>,
<recursive calls to lower routines>,
    <case: root/stem context>,
        <subcase: affix context>,
        <subcase: property context>,
    <action: property ASSIGNment>,
    <result: output structure>,
<repeat SEGMENTation>.

II.1.3 The Architecture of the N-Processor

Like the V-Processor, the N-Processor is a structured program made up of several related procedures. At the lowest level, are independent initial or basic recursive subroutines shared with other programs. These perform the frequent tasks of SEGMENTation, word ASSEMBLY, MATCHing, and root or stem EXTRACTion. Note that root EXTRACTion is a routine shared with the V-Processor, and stem EXTRACTion is a routine that is unique to the N-Processor. At the next level up, are intermediate subroutines for DERIVation from Basic and BP pattern lists, and which implement restrictions on database access and then invoke the said lower routines. At the highest level, are the main procedures that carry out the N-Recognition tasks. These are context-sensitive functions of two types:

1) primary recursive or non-recursive procedures that handle the main parsing, property ASSIGNment, and ambiguity resolution;

2) secondary procedural filters which filter out output from the main procedures, if it eventually violates graphotactic, CASE or TRANSITIVITY constraints. The filters are also different, from those in the V-Processor, in that they do not have access to the N-Database.

For each of the procedures and filters, we have specified in this Chapter a number of free arguments, LOCAL VARIABLES, and objectives, as well as a result to be RETURNed. In parallel to the V-Processor, the N-Processor also has a modular system architecture which provides a flexible and efficient structure for debugging and updating. This modular structure is shown in Figure 33 together with a summary of the main tasks of the N-Processor Modules.

II.2 THE SEARCH PHILOSOPHY OF THE N-PROCESSOR

II.2.1 Converting M-Trees to N-Goal Trees

In Chapter 6, § II.3, we defined a set of MS rules which we represented in the form of Annotated M-Trees. In order to implement these trees as a N-Recognition process, we need to go through an intermediate stage where we convert them into N-Goal Trees. These G-Trees will then allow us to optimize our recognition tasks.

In Chapter 2, § III.1, we defined what a G-Tree is, we discussed its advantages, we described a Conversion Rule for translating an M-Tree into a G-Tree and, and we specified how a G-Tree is to be solved. Figure 34 below shows a G-Tree for the N-Complex in M.S.A. and illustrates the ordered search path followed by the search strategy of the N-Processor.

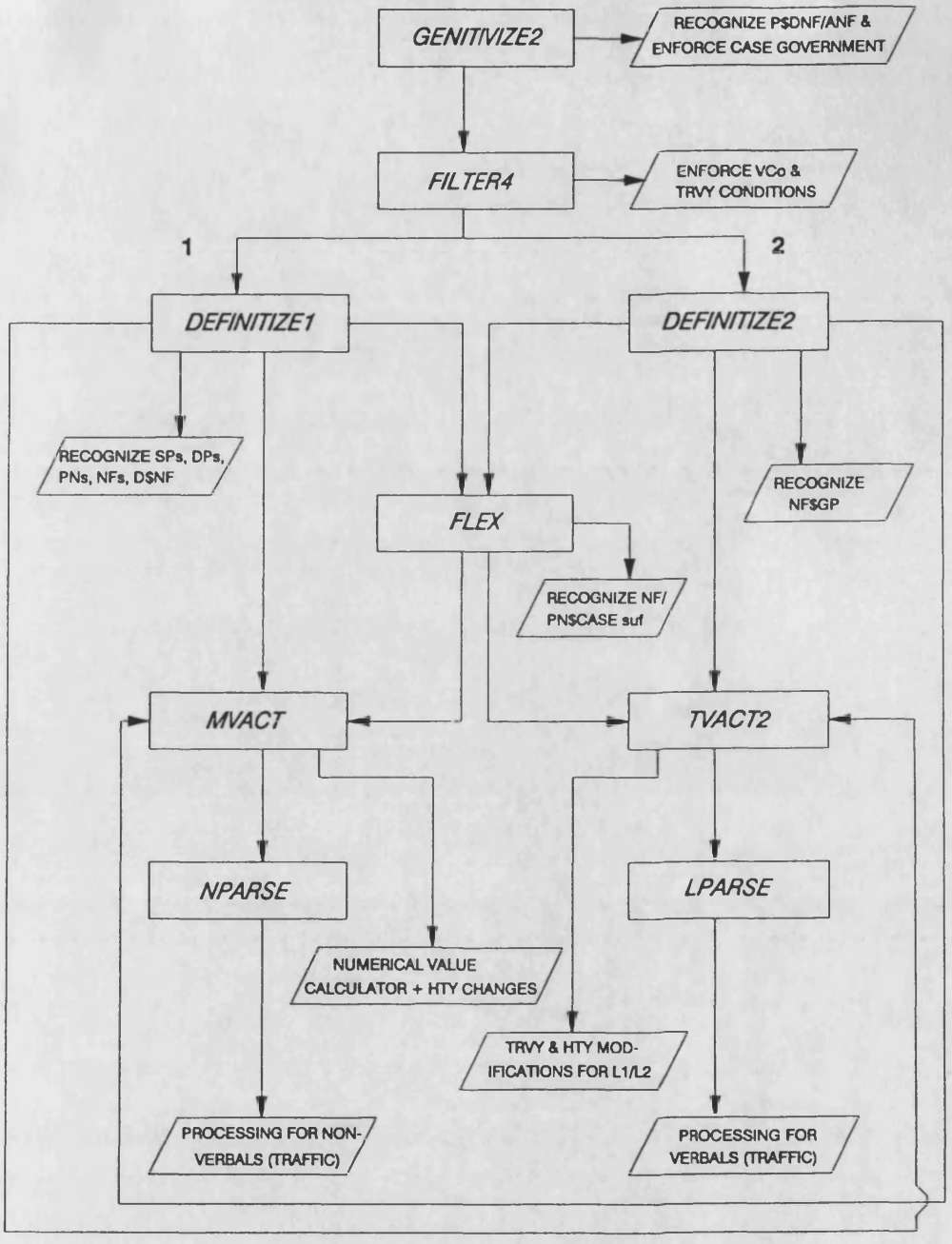


FIGURE 33: The Architecture of the N-Processor in TUNIS1

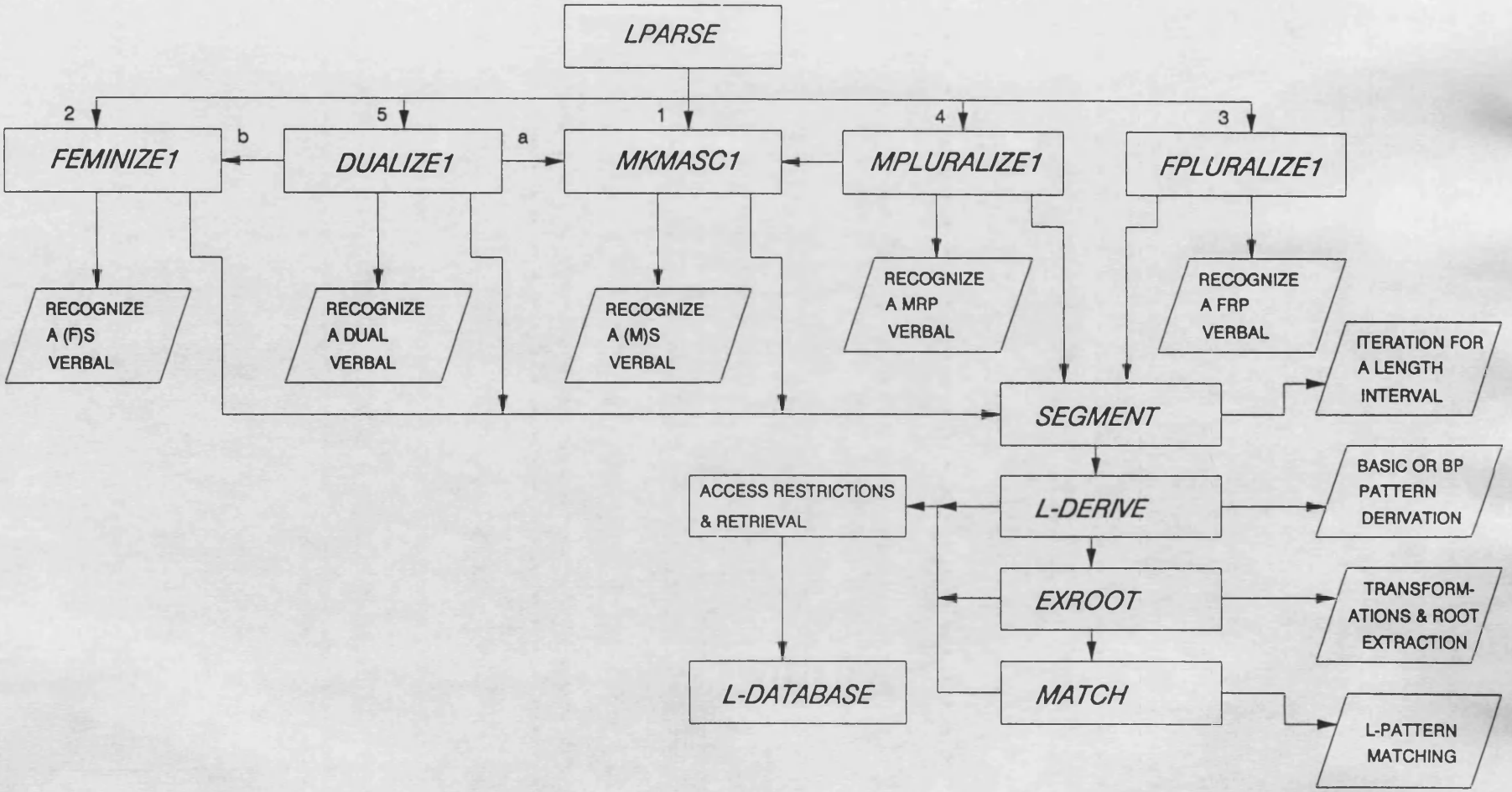


FIGURE 33: The Architecture of the N-Processor in TUNIS1 (Contd)

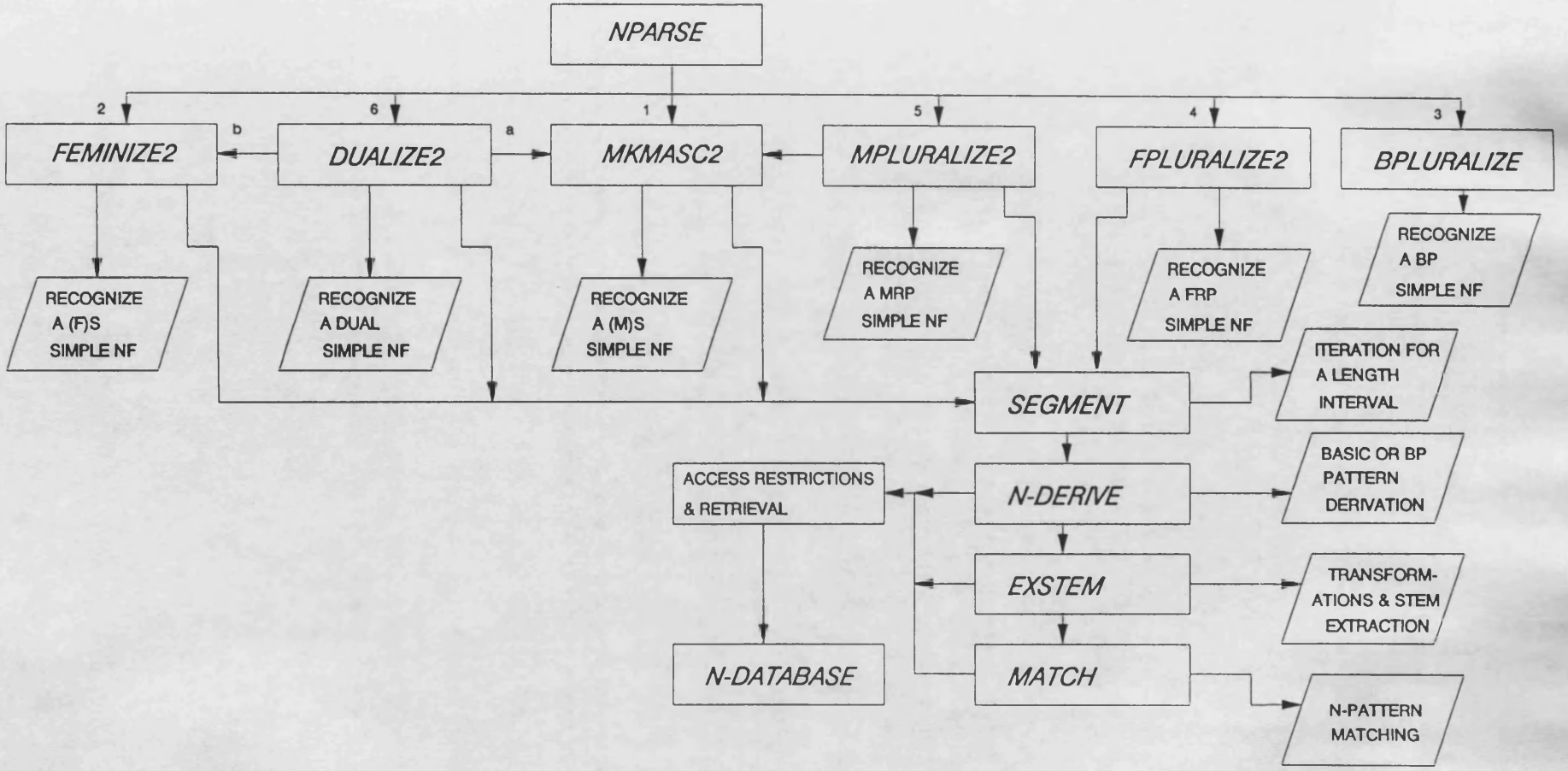


FIGURE 33: The Architecture of the N-Processor in TUNIS1 (Contd)

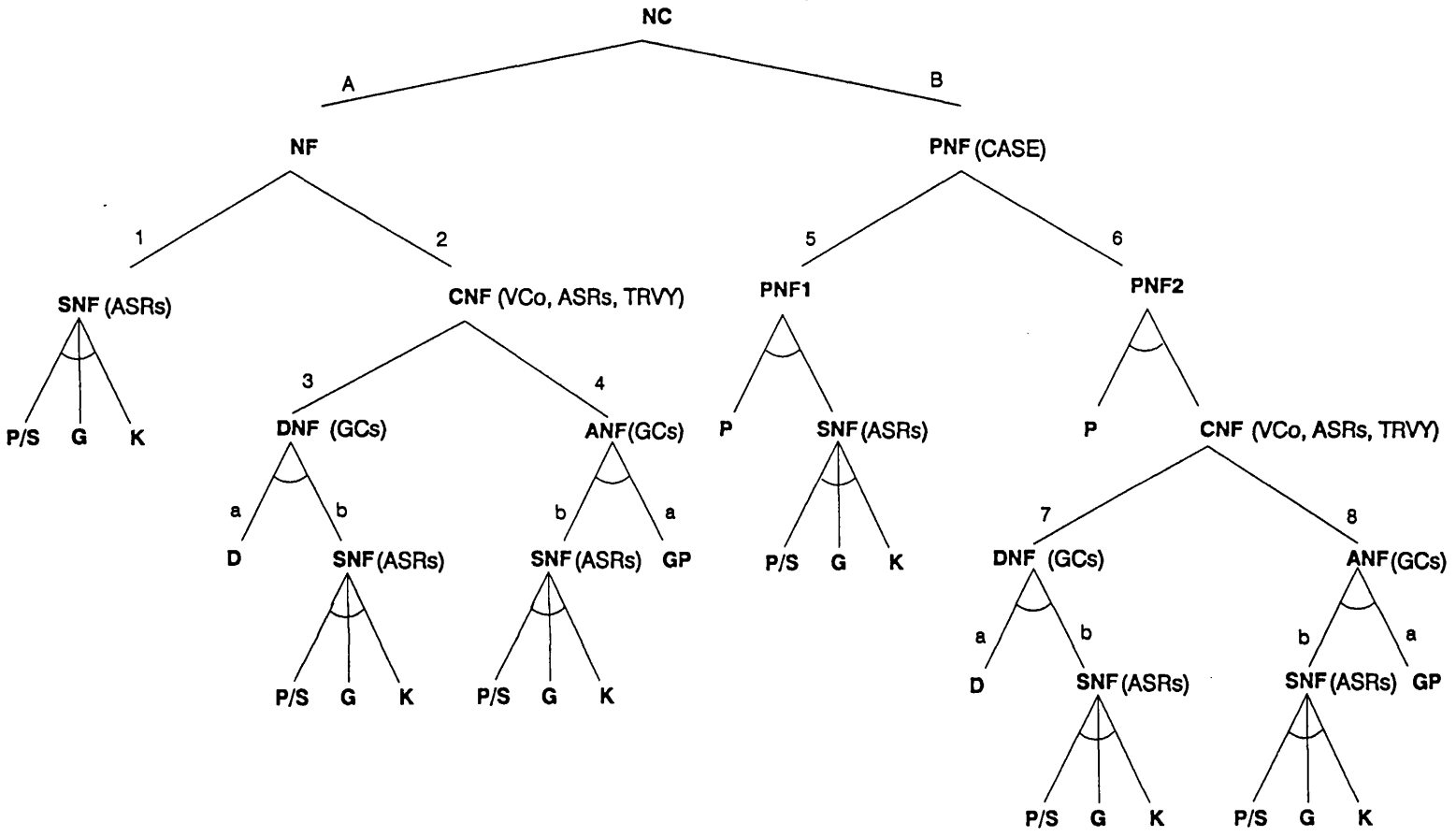


FIGURE 34: Search Paths and Constraints in a G-Tree for the N-Complex in M.S.A.

II.2.2 The Search Strategy and the Flow of N-Parsing

Figure 34 shows a NC G-Tree with a set of *and* and *or* nodes. The question is what is the optimum path for the solution of the NC G-Tree? Our answer can be guided by the principles outlined in Chapter 2, § III.2.3.1,i.e, 1) search for the simplest or minimal most likely concatenations first; and 2) proceed from right-to-left. There is only one optional affix at the end of a N-Inputword, and which is GP. However, there are a possible two to four: QCD$P affixes at the front. We refer to the elements Q and C in Chapters 10 and 11 below.

For the N-Processor, the simplest case is a CASE-inflected Indefinite NF with a possible three segments: an obligatory N-Centre, an optional GENDER suffix, and an obligatory CASE suffix. The N-Processor first searches for the inputword in SPs, DPs, and PNs. These are in fact found by simple database look-up and there is no processing involved. Then, FLEX is invoked where two options are considered: first, MVACT, which invokes the non-Verbal processor: NPARSE, and then, TVACT2, which invokes the Verbal processor: LPARSE. Both of these invoke procedures designed for the recognition of NUMBER/GENDER categories. These procedures are of three types:

a. MKMASC1 and 2, and BPLURALIZE, which look for a possible CASE suffix of LENGTH 1 or 2 at the right of a NF.

b. FEMINIZE1 and 2, and FPLURALIZE, which look for a feminine GENDER suffix of LENGTH 2 or 3 at the right of a NF.

c. DUALIZE1 and 2, and MPLURALIZE1 and 2, all of which look for a NGC suffix of LENGTH 1 to 5 at the right of a NF. They perform iterations for the LENGTH interval and, after identifying the suffix, they then invoke lower procedures such as MKMASC or FEMINIZE.

After identifying the GENDER and NGC suffixes G and K, all the procedures in *a.* to *c.* attempt a DERIVation for the REST of the NF, or its centre, from Basic patterns—for all categories—and from BP patterns, for BP categories. If both MVACT and TVACT2 FAIL, then FLEX strips off a CASE suffix of LENGTH 1 at the right of the NF, looks for the REST of NF in PNs, and if it FAILS, it then recalls MVACT and TVACT2 again and applies them to the REST of the NF.

If FLEX still FAILS, then DEFINITIZE1 is invoked. This time, parsing is resumed in a left-to-right direction, and the next larger concatenation is attempted: D$NF, where D is a prefix of LENGTH [1, 4]. DEFINITIZE1 performs an iteration for this interval. If D is identified, then FLEX is invoked for the REST of the NF and parsing resumes as before. If DEFINITIZE1 FAILS, then DEFINITIZE2 is invoked and another concatenation of the same level is attempted: NF$GP, where GP is a suffix of LENGTH [2, 5]. However, this time parsing

resumes in a right-to-left direction and in a recursive manner using the GP LENGTH interval. If a GP is identified, then FLEX is recalled and parsing continues as above. In some cases, such as for the first-PERSON-singular GP, DEFINITIZE2 does not need to invoke FLEX, but instead recalls MVACT or TVACT2 directly obeying the constraint: *NFKGP where GP = 1 (M, F) S.

At exits from DEFINITIZE1 and DEFINITIZE2, the TRANSITIVITY and Graphotactic Conditions are applied via FILTER4. If FILTER4 RETURNS a NIL result, then GENITIVIZE2 is applied and the largest NF concatenation is attempted: P$CNF. GENITIVIZE2 is a non-iterative procedure which strips off a prefix of constant LENGTH 2. If this prefix is identified in the Bound Prepositions, then FILTER4 is recalled; and if GENITIVIZE2 succeeds, then CASE GOVERNMENT conditions are applied. Figure 35 below illustrates the progress and direction of parsing in the N-Processor.

In parallel to the V-Processor, the N-Processor is then a depth-first exhaustive search without backtracking. Here again, this technique is dictated by the linguistic principles described and by the need for context-sensitive property ASSIGNment and structural and property disambiguation, all of which exclude context-free or blind search methods. However, the search technique in the N-Processor is constrained with the application of the linguistic rules which are implemented as exit conditions on node satisfaction. Note that, here also, there is a process of tree flattening, since Figure 32 shows possible bracketed structures of the form: [(Pr) [(D) Ce (G) K (GP)]]. However, only single or unified categorial types are actually generated. Here are some examples: NN and MD—for Indefinite NFs—DN and AN—for Definite and Annexed NFs—and PDN and PAN, for Prepositional NFs.

For further illustration of the procedures of the N-Processor and using the function MKTRUNK which invokes GENITIVIZE2 as a subprogram, we provide some sample morphological parses by TUNIS1 (cf. Vol. II, App. D, § A).

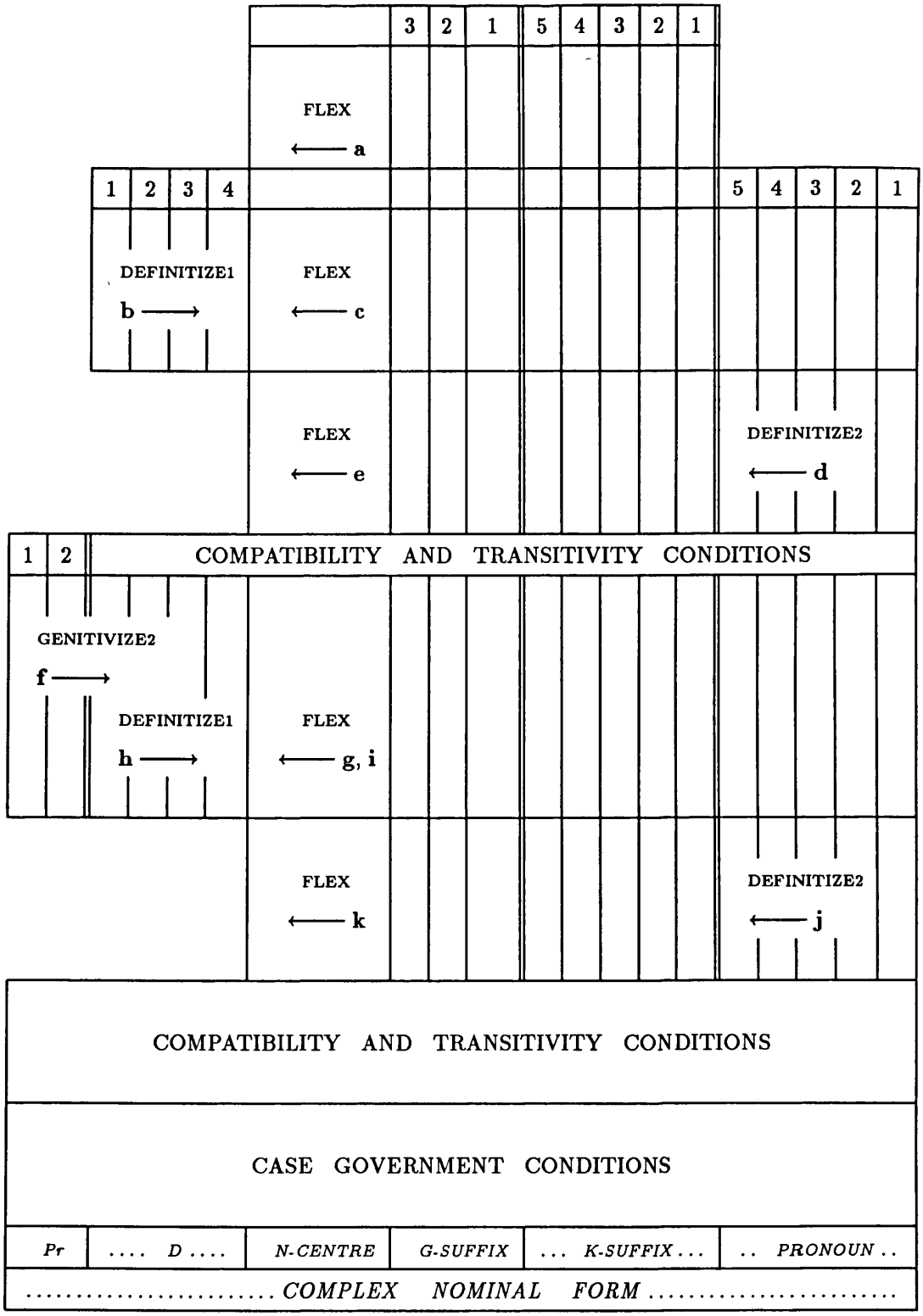


FIGURE 35: A Chart for the Direction of Parsing in the N-Processor

==0==<*****>==0==

Part IV

THE PARTICLE COMPLEX IN M.S.A.

Chapter 8

A FORMAL MODEL OF THE P-COMPLEX

INTRODUCTION

In this Chapter, we analyse the structure of *Particles* in M.S.A. In order to arrive at an adequate description of Particles, we have to seek an optimal typology of these elements, investigate the syntactic functions and requirements implied by their feature values for properties such as CASE or MOOD GOVERNMENT, and examine their graphological and morphological structure. Such investigations should lead us to a satisfactory classification of Particles, and a more precise and sound account of the Particle system in Arabic.

In Section II, we seek optimal generalizations about Particles. We study the specific GOVERNMENT and Graphotactic Conditions of affixation between Particles and other morphemes, as well as allomorphic variation and boundary constraints within Simple and Complex Particle Forms, or PFs.

The statement of these conditions and constraints should yield an adequate and constrained structural grammar based on rewriting rules for PFs. Such a grammar can then be used for the formulation of a formal model of the P-Complex (PC) and its logical structure. This in turn should eventually serve as the basis for the construction of an efficient parser for PCs.

I A LINGUISTIC DESCRIPTION OF THE PARTICLE IN M.S.A.

I.1 A TYPOLOGY OF THE PARTICLE IN M.S.A.

A *Particle* is defined functionally in CRYSTAL, 1980: 258, as "an INVARIABLE ITEM with GRAMMATICAL FUNCTION". This is much the same definition he gives, ibid. 156, to a function word as a "word whose role is largely or wholly grammatical". In the Arabic literature, definitions of *':aloh-arof'*, "the Particle" tend to be semantic and rather trivial. For instance, SHARIYF, 1979: 20, states that the Particle is "that whose meaning only comes into existence as a matter of contrast with other items". We find a similar definition in AL-MAXZUWMIY, 1966: 37, and in DAK ALBAAB, 1982: 56. However, HASSAAN, 1979: 123, gives a somewhat elaborate functional definition of the Particle as "a *categorial class* which denotes a contingency semantic content and the relation it expresses is necessarily dependent on the other elements of the sentence". Such definitions, at least in spirit, have not changed in the last few centuries and since the times of IBNU JINNIY, 1979: 91, which defines the Particle as "that which is allowed to have neither the [inflection] marks of Nouns nor the [inflection] marks of Verbs. It is that which is used solely with a meaning that is in [a word] other than itself.".

We define the Arabic *Particle*, progressively, by describing its syntactic, graphological, and morphological structure, in § I.1.1 and § I.1.2 below; but, for the moment, we understand by *Particle*, in M.S.A., an invariable item which belongs to a *closed-class* set ("not exceeding eighty [words]" (cf. NIϵMA, 1973: Vol. I, 147)) and which cannot, unlike Verbs and Nominals, be derived using a root and a pattern. Further, Arabic *Particles* may occur as affixes (i.e., as bound morphemes) in prefix positions (we have already described Bound Particles such as 'sa', "going to", 'li', "to, for ...", and 'bi', "in, at, with ..."; and, in Chapter 10, we deal with the *Interrogative Particle* ':a', "Q", and the *Coordinative Conjunctions* 'wa' and 'fa', "and") or they may occur as free morphemes.

I.1.1 Syntactic Functions and Requirements of the Particle

Arabic Particles are distinguished according to the type of sentence element they precede—such as Verb or Nominal or both—and as such carry lexical specifications for the GOVERNMENT of their regimen and in some cases for syntactic complementation. Their distribution in relation to the elements they govern is always one of precedence.

The GOVERNMENT of Verbs by Particles consists in lexical specifications for TENSE and MOOD, while the GOVERNMENT of Nominals by Particles consists in lexical specifications for CASE and FLEXION. Particles that govern Verbs can be of several types, such as *Subjunctive* and *Assertive*. Particles that govern Nominals can be *Affirmative*, *Negative* or *prepositional*. However, there are other types of Particle such as *Interrogative Particles* (e.g., 'halo', "whether") which can be followed by Nominals or Verbs alike.

While it is true that Particles are semantically empty CATEGORIES on their own, some Arabic Particles such as *Interrogative Particles* carry prerequisite feature specifications for the type of element that can follow them. So, for instance, 'mano', "who" can be followed only by a human Nominal or a by a Verb that is performed by a human, 'ma|', "what", can be followed only by a non-human or by a Verb that is performed by a non-human, and 'kamo', "how many" has to be followed by a countable Nominal and if a Nominal follows the Negative Particle 'la|', "no, not", it has to be indefinite. In addition, some Interrogative Particles carry a feature value for sentence-marked *MODE* interrogative, as opposed to the unmarked declarative *MODE*, as well as CASE and DEFINITENESS values in some cases.

The syntactic function of Arabic Particles when followed by other elements can range from negation, emphasis, location, and assertion, to subordination, modality, and question. Furthermore, Arabic Particles may have syntactic requirement for the type of complementation they will expect. So for instance, 'kayofa', "how" can be followed by a Nominal, as in 'kayofa :anota?', "how are you?", or by a sentence 'kayofa ja|:a zayodu+?', "how did Zayd arrive?".

All Arabic Interrogative Particles require Verbs that follow them to be in imperfect TENSE and indicative MOOD or in perfect TENSE. Both subjunctive and jussive forms are inhibited in this context. NIϵMA, 1973: Vol. I, 150, like most other Arab grammarians, claims that {qado, sawofa, sa, etc.} "have no governmental effect on the Verb following them". This fallacy is due to the fact that the indicative is the unmarked MOOD, but in a formal grammar such as this and contrary to the usual practice by Arab grammarians, that MOOD value has to be specified if we are to avoid overgeneration.

I.1.2 Graphological and Morphological Structure of the Particle

Each Arabic Particle has a graphologically invariable structure. Its form is a non-derived primitive one which does not have a root-pattern association for dynamic derivation. Morphologically Arabic Particles have free or bound distribution.

As for affixation, Arabic Particles can be affixed to either Free or Bound Particles and Bound Particles can be prefixed to CFs or to NFs (cf. Table 54, below, and Ch. 10, Tables 57–59). In this Chapter, we will cover the various types of Particles as outlined above and as summarized in Table 53 below.

I.1.3 The Arabic Particle System

To illustrate the complexity of the Particle system we can give the example of the primitive *Free Affirmative Particle* ':inna', "indeed" which has to be followed by a definite or an indefinite Nominal that occurs in an Arabic Nominal sentence where the Nominal is in the accusative

PARTICLE	CAT	REGIMEN	GOVERNMENT		EXAMPLE
			TENSE/DFN	MOOD/CASE, FLXN	
Subjunctive	SP	VERB	imperfect.	subjunctive.	'lano' "never"
Jussive	JP	VERB	imperfect.	jussive.	'lamo' "not"
Future	FD	VERB	imperfect.	indicative.	'sawofa' "will"
Assertive	PD	VERB	perfect, imperfect.	indicative.	'qado' "perhaps, already"
Negative	EP	NOMINAL	indefinite.	accusative, & not fvm.	'la|' "not"
Affirmative	AP	NOMINAL		accusative.	':inna' "indeed"
Preposition	Pr	NOMINAL		prepositional	'baʕoda' "after"
Interrogative	HQ	VERB, NOMINAL	perfect, imperfect.	indicative/ neutral of CASE.	'mano' "who"
	JQ				'ma|' "what"
	Q2				'halo' "whether"
	QP				':ayona' "where"

TABLE 53: Categorial Types and GOVERNMENT Conditions for the Particle in M.S.A.

CASE but is of unspecified FLEXION type. This Particle has priority front position and precedes all the other elements, except for a small subset of bound prefixes such as ':a', "Q", 'wa', "and", and 'li', "to, for ...", which have higher precedence or priority. Hence, this Particle itself can be affixed to such elements or to others like Accusative Pronouns. ':inna', "indeed" fulfils the syntactic function of affirmation. It carries the semantic content "it is true that (*sentence proposition*)", and the morphological CATEGORY *AP*, for Affirmative Particle.

Table 54 below shows a multidimensional representation for the Arabic Particle system.

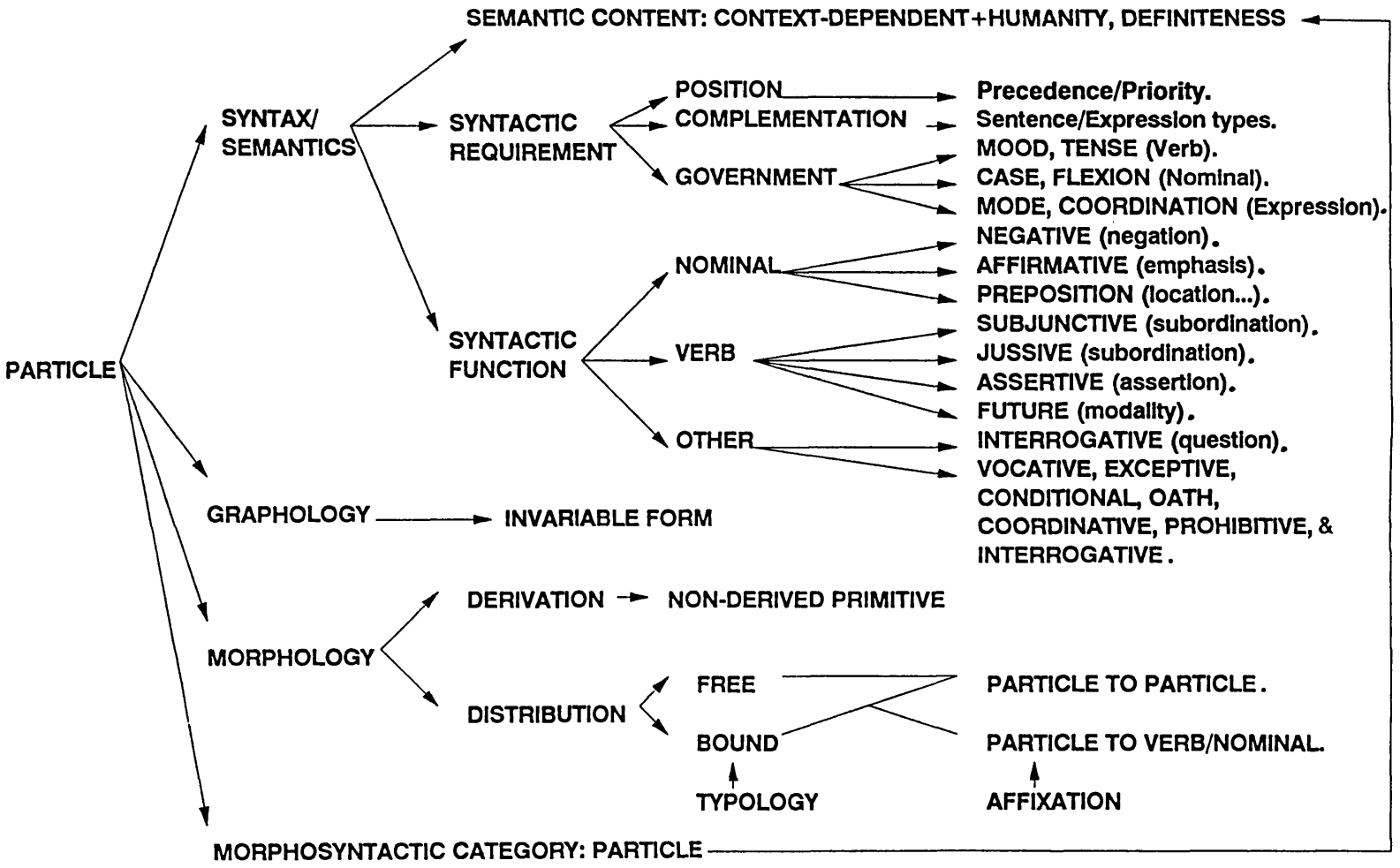


TABLE (54): The Arabic Particle System

I.2 SCOPE OF THE P-ANALYSIS

In this study, we have covered representative but not exhaustive examples of Particles from amongst each of the types of Particle as listed above. Because of time and space constraints, we have not covered other types of Particles such as *Vocative, Exceptive, Prohibitive, Conditional,* and *Oath Particles.* In addition, some Particles have multiple dynamic functional roles in that they can precede Nominals in one role and Verbs in another. For instance, 'la|', "no, not", when preceding a Nominal is a Negative Particle, and when preceding a Verb can be a Prohibitive Particle. The solution adopted here was to take one role as the default, and leave other functions to be distinguished as contextual ones, at the level of syntactic parsing.

The above treatment of a subset of Arabic Particles illustrates a descriptive and classificatory technique which can be *extended* to the analysis of Arabic Particles as a whole.

II A FORMAL ACCOUNT OF THE PARTICLE IN M.S.A.

II.1 CONDITIONS OF AFFIXATION IN SIMPLE PFs

We cannot make pattern-root generalizations about Particle Forms, or PFs; however we can specify particular GOVERNMENT and Graphotactic Conditions for the affixation of different Particles to other morphemes. These conditions are parallel to those observed in CFs and NFs.

II.1.1 GOVERNMENT Conditions and Graphotactic Transformations Governing Simple PF Boundaries

Free Prepositions (Prs) can be affixed to GPs and govern them in oblique CASE, while *Affirmative Particles* (APs) can be affixed to CPs and govern them in accusative CASE. This affixation is also governed by simple boundary transformations which force allomorphic variation in the form of Prs or APs. This variation is described immediately below. First, we define a Simple PF as an AP prefixed to a CP, or as a JQ, AQ, QP, PD, FD, ED; and we define a PGP as a Pr prefixed to a GP.

a. $\forall$ Pr & GP / [ ] Pr$GP,

i. *if* >2 & >1 (Pr) = 'iy' & GP = 1 (M, F) S $\Longrightarrow$ GP = 'ya'.

ii. *if* >2 & >1 (Pr) = 'ay' & GP = 1 (M, F) S $\Longrightarrow$ GP = 'ya'.

iii. *if* >2 & >1 (Pr) = 'ay' & GP $\neq$ 1 (M, F) S $\Longrightarrow$ Pr $\rightarrow\!\!\!\!\!\rightarrow$ Pr$<o>.

iv. *if* >1 (Pr) = <a> & GP = 1 (M, F) S $\Longrightarrow$ <a> $\rightarrow\!\!\!\!\!\rightarrow$ Ø & GP = 'iy'.

v. *if* >1 (Pr) = <o> & GP = 1 (M, F) S, D, P $\Longrightarrow$ <o> $\rightarrowtail$ Ø & GP $\neq$ 'ya'.

b. $\forall$ AP & CP / [ ] AP$CP,

i. $\forall$ AP $\Longrightarrow$ CP $\neq$ 'ya'.

ii. $\forall$ AP $\Longrightarrow$ >1 (AP) = <a> $\rightarrowtail$ Ø where CP = 'iy'.

The rules in *a.* govern the affixation of Prs to GPs and state that "forall PGPs if Pr ends in 'iy', or 'ay' and GP is in the first-PERSON singular, then it must have the form 'ya'. If Pr ends in 'ay' and GP is not in the first-PERSON singular, then <o> must be inserted at the end of Pr. If Pr ends in <a> and GP is in the first-PERSON singular, then <o> must be deleted and GP must have the form 'iy'. If Pr ends in <o> and GP is in the first-PERSON singular or plural, then <o> is deleted and GP cannot have the form 'ya'". This rules out sequences such as '*fiy$iy', '*fiy$niy', '*laday$iy', '*laday$niy', '*:ilay$himo', '*mino$ya', '*mino$niy', '*mino$na|', '*qabola$ya', and '*qabola$iy'. The legal forms are 'fiy$ya', "in me", 'laday$ya', "with me", ':ilayo$himo', "to them", 'min$niy', "from me", 'min$na|', "from us", and 'qabol$iy', "before me".

The rules in *b.* govern the affixation of APs to CPs and state that "forall PFs, CP must never have the form 'ya' and the final character of AP, which is <a>, must be deleted where CP has the form 'iy'". This rules out '*:inna$ya', '*:anna$ya', '*:inna$iy', and ':*anna$iy'. The legal forms are ':inn$iy', ':inna$niy', ':ann$iy', ':anna$niy', "indeed, I am". The allomorphic variation in Pr and AP forms is given in Table 55 below.

II.1.2 Graphotactic Compatibility Conditions Governing Simple PF Boundaries

The boundaries of PFs and PGPs are also governed by Graphotactic Conditions of compatibility between the Particle or Preposition and the enclitic Accusative or Genitive Pronoun, whether the Preposition is Free or Bound. These are parallel conditions to those in Chapter 4, II.4.3.*c.* and Chapter 6, II.1.6.2.*d.vii.*, although there are slight differences in context of rule application as follows:

a. $\forall$ P & TP / [ ] P$TP where P is a Particle or a Preposition, TP is a GP or a CP, and TP $\neq$ (F) S,

i. *if* >1 (P) = {i/y}, or

ii. *if* >1 (P) = <o> & >2 (P) = <y> $\Longrightarrow$ <2 (TP) = <i>.

iii. *if* >1 (P) = <o> & >2 (P) $\neq$ <y>, or

iv. *if* >1 (P) = <a> $\Longrightarrow$ >2 (TP) = <u>.

PARTICLE TYPE	PREFIX FORM		CONTEXTUAL ENCLITIC PRONOUN VALUES (FOR NGP)
	DEFAULT	CONTEXTUAL	
Pr (variable)	laday ɛalay :ilay	ladayo ɛalayo :ilayo	GP ≠ 1 (M, F) S.
	mino ɛano	min ɛan	GP = 1 (M, F) S, D, P.
	ɛinoda fawoqa baɛoda tah-ota :ama|ma qabola wara|:a maɛa	ɛinod fawoq baɛod tah-ot :ama|m qabol wara|: maɛ	GP = 1 (M, F) S.
Pr (constant)	fiy		Any.
AP (variable)	:inna :anna	:inn :ann	CP = 1 (M, F) S.

TABLE 55: Particle Form Variation with TP Values

The graphotactic compatibility rules in *a.* allow the free affixation of the feminine-singular Pronoun to other Particles, but stipulate that all other Pronouns should have an allomorphic variant that is compatible in vocalic constitution with the Particle it is affixed to. Here again such conditions are motivated by pronunciation rules. They rule out the following sequences: '*fiy$hu', '*:ilayo$hu', '*mino$hi', '*la$hi', and '*:inna$hi'. The legal forms are 'fiy$hi', "in him", ':ilayo$hi', "to him", 'mino$hu', "from him", 'la$hu', "for him", and 'inna$hu', "indeed, he".

II.1.3 Affix Selection Rules Governing the Affixation of Bound Prepositions to Simple PFs, GPs, and Other Particles

Besides compatibility conditions, the affixation of the Bound Prepositions: 'la', "to", 'li', "to, for ...", 'bi', "in, at ...", and 'ka', "as", to PFs, GPs, and other Particles is subject to Affix Selection Rules which are motivated by pronunciation rules, but which are sometimes seemingly idiosyncratic. These constraints can be expressed as follows:

a. There are general constraints governing the following sequences:

i. *P$GP is not allowed where P $\neq$ {bi, la}, "in", "at".

ii. *PPX$ is not allowed where PX is a Bound Particle attached to any other morpheme X.

iii. PJQ, PHQ & P$QP are allowed only where P $\neq$ 'la', "to" & QP = 'ma|d-a|', "what".

iv. P$EP is allowed only where P = 'bi', "in, at ...".

v. P$FD & P$PD are allowed only where P = 'la', "to".

The rules in *a.* respectively inhibit the following sequences, for which there are no legal counterparts.

- '*ka$hu', '*ka$ya', '*ka$iy', and '*li$hu' (by rule *i.*).
- '*libika', '*bilaka', and '*bilizayodi+' (by rule *ii.*).
- '*la$ma|da|', '*la$mano', and '*la$kayofa' (by rule *iii.*).
- '*li$la|', '*ka$la|', and '*la$la|' (by rule *iv.*).
- '*bi$sawofa', and '*ka$qado' (by rule *iv.*).

Note that rule *a.ii.* also contradicts NIℇMA's 1973: Vol. I, 150, claim, quoted above, of free sequencing between 'sa', "going to", 'sawofa', "will", etc., and other morphemes.

b. In addition, there are more specific constraints governing the following sequences:

i. $\forall$ P & GP / [ ] P$GP,

- *if* P = {li/bi} & GP = 1 (M, F) S $\Longrightarrow$ <1 (P) = <i> $\rightarrowtail$ Ø & GP = 'iy'.
- *if* P = 'la', "to" & GP $\neq$ 1 (M, F) S $\Longrightarrow$ *P$GP.
- *if* P = 'la', "to" & GP = 1 (M, F) S $\Longrightarrow$ P$GP.

ii. $\forall$ P & AP / [ ] P$AP $\Longrightarrow$ P $\neq$ 'la', "to" & <2 (AP) = <a>.

The rules in *b.* respectively inhibit the following sequences:

- '*la$iy', '*la$ya', '*la$niy', '*bi$iy', '*la$hi', and '*la$himo', (by rule *i.*).
- '*la$:anna', '*li:inna', and '*bi:inna', (by rule *ii.*).

The legal sequences are 'l$iy', "to me", 'b$iy', "with me", 'la$hu', "to him", 'la$humo', "to them", 'li$:anna', "because, indeed, ...", and 'bi$:anna', "that, indeed, ...".

II.2 ANNOTATED MS RULES FOR COMPLEX PFs AND PGPs

The above analysis of Particle affixation implies a formal morphological grammar, or MS rules, for rewriting PFs and PGPs. Such rules can be augmented with annotations, as was the case with CFs and NFs. The incorporation of the GOVERNMENT and Graphotactic Conditions of affixation in the expression of morphological rules will ensure the generation of "all and only" valid sequences and inhibit the generation of invalid sequences such as those in II.1.3.*a.* and *b.*

We have already stated, in § II.1.1 above, that a Simple PF is an Affirmative Particle AP which is prefixed to a GP, and that a PGP is a Pr prefixed to a GP, or an obligatory JQ, HQ, QP, PD, FD or EP. The rules in § II.1.3 imply Complex PF (CPF) sequences and which we define as Bound Prepositions prefixed to Simple PFs.

We can formulate a CSG incorporating annotations that represent the various constraints and conditions described above and specifying the grammatical morphological sequences as in *a.* below:

a.i.	CPF	⟶	(P [+CASE, +ASRs]) PF [+CASE, +ASRs]
a.ii.	PF	⟶	AP [+VCo, +CASE, +ASRs] (CP [+VCo, +CASE, +ASRs])
a.iii.	PF	⟶	JQ/HQ/QP/PD/FD/EP
a.iv.	PGP	⟶	Pr [+VCo, +CASE, +ASRs] (GP [+VCo, +CASE, +ASRs])
a.v.	PGP	⟶	P [+VCo, +CASE, +ASRs] GP [+VCo, +CASE, +ASRs]
a.vi.	P	⟶	{li/la/bi/ka} [+VCo, +ASRs, +OBL]
a.vii.	Pr	⟶	{fiy/laday/ladayo/ɛalay/ɛalayo/:ilay/:ilayo/mino/ɛano/ ɛinoda/fawoqa/baɛoda/tah-ota/:ama|ma/qabola/ wara|:a/maɛa} [+VCo, +OBL]
a.viii.	TP	⟶	{niy/iy/ya/na|/ka/ki/kuma|/kumo/kunna/hu/hi/ha|/huma|/ hima|/humo/himo/hunna/hinna} [+VCo, +ASRs]
a.ix.	AP	⟶	{i:nna/:anna} [+ASRs, +acm]
a.x.	JQ	⟶	'ma|'
a.xi.	HQ	⟶	'mano'
a.xii.	QP	⟶	{ma|;d-a|/kayofa/matay/:ayona}
a.xiii.	PD	⟶	'qado'
a.xiv.	FD	⟶	'sawofa'
a.xv.	EP	⟶	'la|'

Grammar *a.* provides a formal generation device for Simple and Complex PF and PGP sequences with optional and obligatory elements.

Rules *a.i.* allows the generation of a P$PF *IFF* the CASE and ASRs of PF are equal to

those of P, where P is optional. Rule *a.ii.* generates a Simple PF, which is an obligatory AP followed by an optional CP, *IFF* the CASE, VCo, and ASRs of CP are equal to those of AP. Rule *a.iii.* generates a PF as an obligatory P attached to an obligatory single element being a JQ, HQ, QP, PD, FD, or EP. Rules *a.iv.* and *a.v.* allow the generation of a Simple PGP in two ways: either as an obligatory Free Preposition followed by an optional GP, provided their VCo, CASE, and ASRs match, or as an obligatory Bound Preposition followed by an obligatory GP, provided their VCo, CASE, and ASRs match. Rules *a.i.* to *a.v.* are, thus, non-terminal rules; while rules *a.vi.* to *a.xv.* are terminal, or lexical, rules requiring that each lexical item specify its own VCo, CASE, and ASRs. For instance, 'li', "to, for, ..." would specify that an AP following it must have, as its second character, the vowel <a> and not <i>.

Using the Annotated Grammar in *a.* we can generate Annotated M-Markers of the type in Figures 36–38 and we can generate Particle sequences of the type in *b.* below:

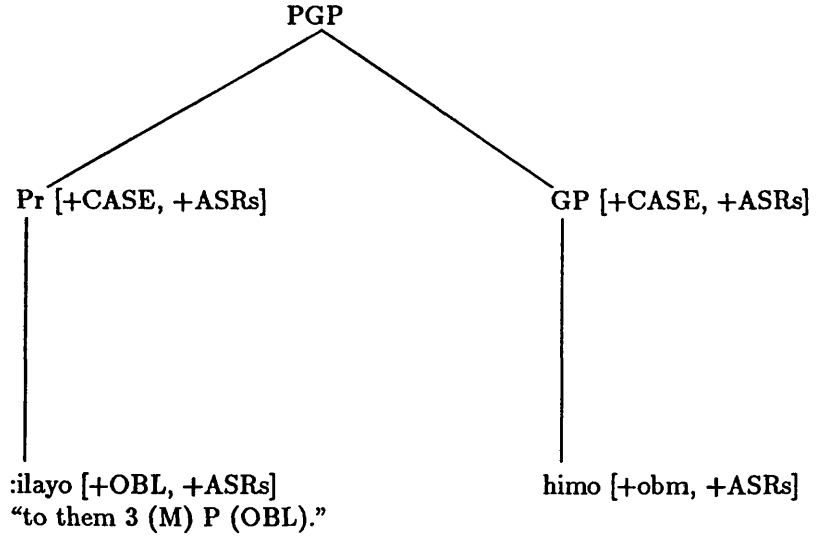


FIGURE 36: An Annotated M-Marker for a PGP of Type 1

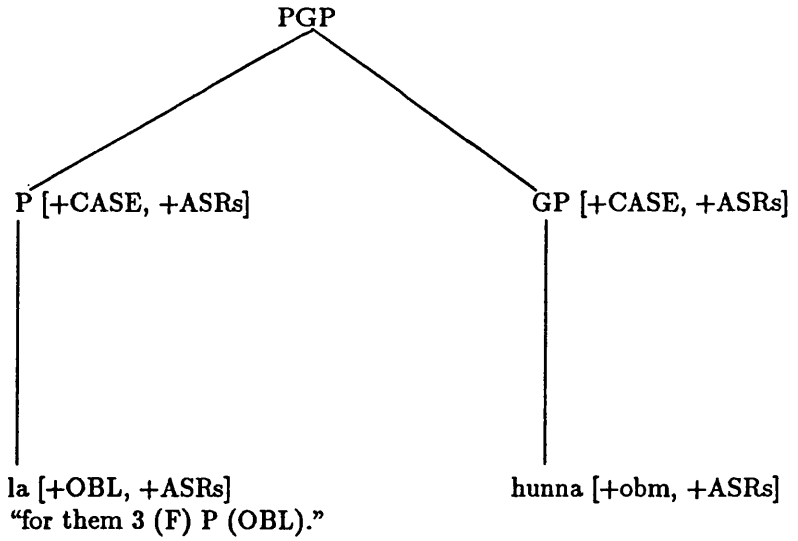


FIGURE 37: An Annotated M-Marker for a PGP of Type 2

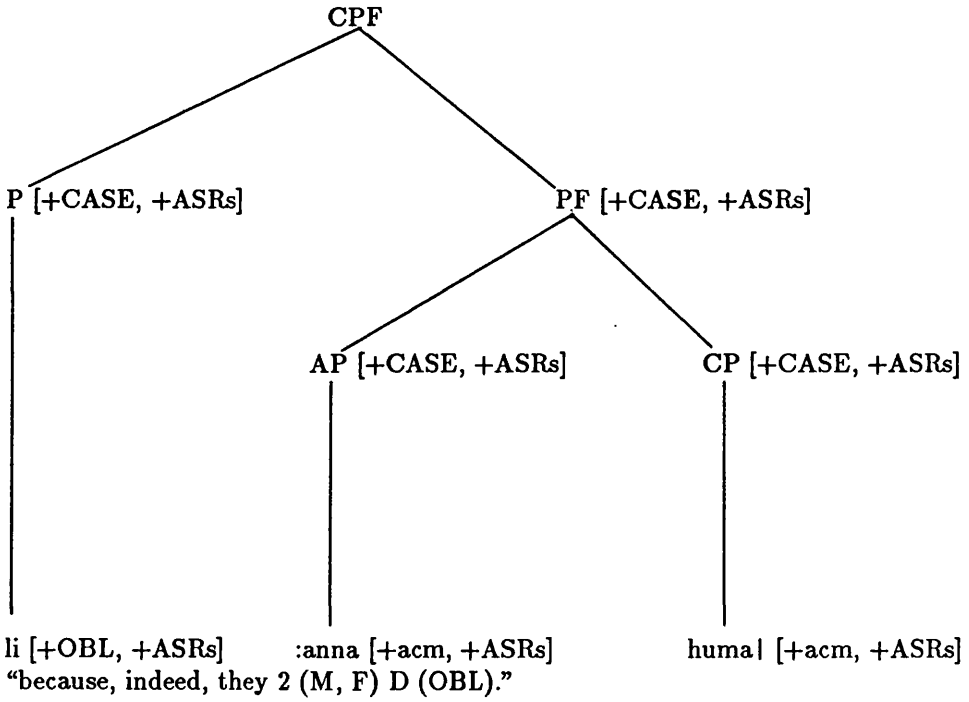


FIGURE 38: An Annotated M-Marker for a Complex PF

b.i.	'maʕa', "with":	Pr
b.ii.	':inna', "indeed":	AP
b.iii.	':anna', "indeed":	AP
b.iv.	'maʕa$hu', "with him":	P$GP
b.v.	'ʕan$na|', "on us":	P$GP
b.vi.	'min$niy', "from me":	P$GP
b.vii.	':ilay$ya', "to me":	P$GP
b.viii.	'baʕod$iy', "after me":	P$GP
b.ix.	'fiy$ya', "in me":	P$GP
b.x.	'ladayo$himo', "with them":	P$GP
b.xi.	':inna$niy', "indeed, I":	AP$CP
b.xii.	':anna$ha|', "indeed, she":	AP$CP
b.xiii.	'li$:anna$ka', "because, indeed, you":	P$PF
b.xiv.	'bi$:anna$na|', "that, indeed, we":	P$PF
b.xv.	'la$sawofa', "really, it will...":	P$PF
b.xvi.	'li$ma|d-a|', "why":	P$PF
b.xvii.	'li$mano', "to whom":	P$PF

==0==<*****>==0==

Chapter 9

PROCESSING THE P-COMPLEX

INTRODUCTION

In Chapter 8, we provided structural descriptions of P-Complexes. In Chapter 9, we examine how to convert those descriptions to formal specifications with requirements for LENGTH, boundary, and lexical conditions. We then convert these specifications, in turn, to an efficient program for parsing Simple and Complex PF and PGP groups and which we will call the *P-Processor*. Before that, we have to build an efficient P-Database with a set of automatic dictionaries and property lists and define restrictions on dictionary look-ups in order to reduce search space.

The progress of each procedure in the P-Processor has to be spelled out and illustrated, and the relationship between its various modules made explicit. The translation of the specifications into constraints and conditions has to be explained and the method for filter control detailed.

Finally, we provide a synthesis of Chapters 8 and 9, marrying the linguistic to the computational analyses, bringing together the architecture of the P-Processor and its search philosophy, joining the M-Grammar for PFs with the structure of G-Trees, and finally, reconciling the direction of flow of parsing to the linguistic principles adopted.

I CONTENTS OF THE P-COMPONENT

I.1 THE P-DATABASE

I.1.1 Data Representation, Data Access, and Property ASSIGNment

In order to attain an adequate representation for morphological data representation and access in the P-Processor and in the construction of a P-Database with efficient storage, ASSIGNment, retrieval and manipulation, we shall take advantage of the techniques used by the V- and N-Processors. Namely, we shall assume such functions as SETQ, FLUID, MEMQ, LENGTH, LAST, EQUAL, XN, CAR, and CADR for the definition of dictionaries with identifiers and lexical entries. We shall use such dictionaries in the expression of dictionary-dependent rules. Further, we shall make use of LISP facilities for property ASSIGNment harnessing the functions PUT, MAPC, and GET to ASSOCIATE feature labels and values with lexical entries.

The ASSIGNment of such properties follows the descriptions of Particle properties as specified in Chapter 8. In the P-Database, this ASSIGNment is as follows:

a. Unique values ASSIGNed initially as default and transferred at run time as fixed or as dynamic values depending on context. These unique values are ASSIGNed as follows:

- CASE GOVERNMENT: accusative, oblique.
- MOOD GOVERNMENT: subjunctive, jussive.
- DEFINITENESS: definite.
- BINDING: bound.
- MODE: interrogative.
- CATEGORY: Pr, JP, SP, AP, EP, JQ, HQ, QP, Q2, FD, PD, LD, AD.

The property "bound" is ASSIGNed to the prepositional allomorphs ':ilayo', "to, until", 'ɛalayo', "on", and 'ladayo', "with" which occur only as bound. Thus, they contrast with their allomorphic variants ':ilay', 'ɛalay', and 'laday', which have the same respective meanings but occur as free.

b. Neutral values ASSIGNed only for CASE as a property of Interrogative Particles using the neutral CASE label " ncp".

For instance, 'mano', "who" occurs with a CASE value nominative, accusative, or oblique. The ASSIGNment of a neutral CASE value to 'mano' ensures its flexible analysis, at the level of syntactic processing, in the same way that the use of neutral values, as described in Chapter 5, § I.1.2, and Chapter 7, § I.1.1, leads to more flexibility at the level of morphological processing.

There are no homonymic values involved in the P-Database except those already covered

elsewhere (in Ch. 5, § I.1.2, and Ch. 7, § I.1.1) for Bound Particles. Neither are there any root-pattern ASSOCIATions involved.

I.1.2 The Structure of the P-Database

The P-Database is a superset which consists of:

- A set of dictionaries with identifiers and lexical entries for Particles and Adverbs.
- A set of properties with initial property values defined on those lexical entries.

The set of dictionaries and PLISTS thus defined constitutes the P-Database. These are listed in Volume II, Appendix B, § G, as follows:

a. A Dictionary of Prepositions containing the set of Free Prepositions carrying values for CATEGORY and CASE GOVERNMENT.

b. A Dictionary for the rest of the Free Particles containing the Jussive, the Subjunctive, the Affirmative, the Negative, the Assertive, the Future, and the Interrogative Particles including the yes-no Question Particle 'halo', "whether", the non-human, human, and other Interrogative Particles.

c. A Dictionary for the rest of the Bound Particles including the Bound Interrogative Particle and the Coordinative Conjunctions which are described below, in Chapter 10.

d. A Dictionary of Place and Time Adverbs.

Each morpheme in these dictionaries carries its own values, for properties such as CASE and MOOD GOVERNMENT, DEFINITENESS, MODE, CASE, and CATEGORY; and all the dictionaries in *a.* through to *d.* are included in a global superset called a *Function Word Table* (FWT) which is accessed in *direct* look-up, viz., a search that does not involve processing. This is explained in Chapter 11.

I.2 THE P-PROCESSOR

I.2.1 Initial Closed Subroutines

The P-Processor needs to avail itself of the same independently-defined closed subroutines to carry out primitive operations as specified in Chapter 3, § II. Such operations will handle tasks like TRANSCRIPTION, SEGMENTation, word ASSEMBLY, character PICKing, COPYing, SUBSTITUTion, and DELETion. Thus we shall make the same assumptions as those in Chapter 6, § I.2.1.

DESCRIPTION TYPE	an abstract composite template structure.
STRUCTURE	*BASIC STRUCTURE:* either sequences of: an obligatory Free Preposition slot, and an optional Pronoun slot, or: obligatory Bound Preposition and Pronoun slots, for PGP; or sequences of: optional Bound Preposition and Pronoun slots, and an obligatory Affirmative Particle slot, or: an optional Bound Preposition, and an obligatory JQ, HQ, QP, PD, FD or EP slot, for CPFs. *CONCATENATION ORDER:* given as: Pr + (GP), P + GP, (P) + (AP) + (CP), or (P) + JQ/HQ/QP/PD/FD/EP. *STRUCTURAL TYPES:* (a) *Prepositional*: PGP. (b) *Affirmative*: AP. (c) *Referential PF*: AP$CP. (d) *Affirmative CPF*: P$AP. (e) *Referential Affirmative CPF*: P$PF.
CONSTRAINTS	*GRAPHOTACTIC CONDITIONS:* include boundary conditions such as Affix Selection Rules, and Vocalic Compatibility; as well as CASE GOVERNMENT. *PGP/CPF LENGTH:* (a) *if* TP then LENGTH (CPF/PGP) > threshold (LENGTH (TP)); and (b) *if* P then LENGTH (CPF/PGP) > LENGTH (P); and (c) *if* P & TP then LENGTH (CPF/PGP) > LENGTH (P) PLUS threshold (LENGTH (TP)). *TP LENGTH:* L = [2, 5]. *P LENGTH:* L = [2]. *PGP/CPF BOUNDARIES:* (a) *if* TP then >1 (CPF/PGP) = >1 (TP), and >1 (TP) = {a/i/o/u/y/l}; and (b) *if* P then <1 (CPF/PGP) = <1 (P), and <1 (P) = {l/b/k}; and

TABLE 56: A Formal Specification for Simple and Complex PFs and PGPs

CONSTRAINTS	(c) *if* Pr then <1 (PGP) = <1 (Pr), and <1 (Pr) = {:/b/ε/f/l/m/q/t/w}. *LEXICAL SPECIFICATIONS:* (a) *if* PGP then, CASE (GP) = CASE (P/Pr), and ASRs (GP) = ASRs (P/Pr); and (b) *if* AP & CP then, CASE (CP) = CASE (AP), and ASRs (CP) = ASRs (AP); and (c) *if* P & AP or PF, and PF = AP$CP then, CASE (PF) = CASE (P), and ASRs (PF) = ASRs (P); and (d) *if* P & PF, and PF ≠ AP$(CP) then, ASRs (PF) = ASRs (P).
ROLE	provide a template for the realization of PGPs and CPFs.

TABLE 56: A Formal Specification for Simple and Complex PFs and PGPs (Contd)

I.2.2 The Main P-Recognition Procedure: GENITIVIZE1

From the formal account given of Particles, in Chapter 8, ensues a formal specification for Simple and Complex PFs and PGPs such as in Table 56 above.

We can now define *GENITIVIZE1*: the main P-Recognition Procedure based on the specifications of Table 56. GENITIVIZE1 is a context-sensitive iterative procedure which, given an inputword x as an argument, attempts to identify it as a PF that is an AP$CP sequence or a PGP. In order to accomplish this task, GENITIVIZE1 first checks if the CPF or PGP satisfy the entry conditions of conforming to the initial and final boundary requirements and, if this is so, it proceeds right-to-left, stripping off TPs of a LENGTH interval: [2, 5], starting at the threshold and incrementing the index by 1 each time a traverse down the loop FAILS. The loop is entered upon identifying a final suffix among GPs or CPs, and this suffix is inspected to see if it is in the first PERSON.

The recognition of a first-PERSON Pronoun triggers off the application of simple boundary transformations to the prefix as well as the implementation of the CASE GOVERNMENT and Graphotactic Conditions of Particle affixation to Pronouns as detailed in Chapter 8. After transformations, the modified prefix is searched for and, if it is found among the Free or Bound Prepositions or the Affirmative Particles, then GENITIVIZE1 proceeds to PASSON the property heritage for NGP, DEFINITENESS, REFERENCE, CASE MARK, HUMANITY, and CATEGORY from the pronominal suffix to x, and ASSIGN to the Pronoun the CASE, required by the prefix. It then RETURNS a result which is a list of the Preposition or Affirmative Particle

and the Pronoun. If the Pronoun is not a first-PERSON Pronoun, then if the prefix is in the set of APs, Free or Bound Prepositions and satisfies the constraints on affixation, viz., if ASRs obtain, then property ASSIGNment is initiated as above, and the result is RETURNed. If the suffix is not identified in the Pronouns, then the index is incremented and the iterative process is resumed and continued until a positive result is obtained or the upperbound is reached and the procedure is aborted. Thus, GENITIVIZE1 does not invoke any lower routines such as, DERIVation or other recognition procedures, since it does not have to process a centre, its main tasks being achieved through dictionary look-up after SEGMENTation. Figure 39 below represents GENITIVIZE1.

I.2.3 Graphotactic Filtering and Complex PF and PGP Recognition

In Chapter 8, § II.1, we outlined ASRs and Graphotactic Conditions of compatibility governing the boundaries of Simple and Complex PFs and PGPs. The task of *FILTER5* is primarily to ensure the application of these conditions to output structures from GENITIVIZE1 and in accordance with the specifications of Table 56. This is carried out by applying, to these output structures, declarative context-sensitive rules in order to filter out possible Particle sequences that do not satisfy the affixation constraints. Here, FILTER5 distinguishes Referential Particle Forms with collocutive Pronouns from those with exlocutive Pronouns as the former have no allomorphic variants whereas the latter do.

However, FILTER5, like FILTER2, also performs a processing task and in this respect is not only a filter but also a recognition procedure. This recognition involves the SEGMENTation of the input PF into a prefix of LENGTH 2 and a suffix, as well as the left-to-right search for the prefix in the set of Bound Prepositions and for the suffix in the set of Nominal and Verb Particles: {JQ, HQ, QP, PD, FD, EP, and AP}. Such sequences (of Bound Prepositions and Particles) were not covered by GENITIVIZE1 since it was an iterative procedure designed for Pronoun stripping and since the said sequences do not attach to TPs. If the suffix is not found in the set of Particles listed above, then it is checked to see if it is a 1 (M, F) S GP with the requirement that the final character of the prefix must be DELETed as specified in II.1.3.*b*. If the suffix is not identified as a GP, then GENITIVIZE1 is invoked on the suffix and, if it is a Simple PF and ASRs obtain on it, then the procedure is successful. The FAILure of FILTER5 causes an error message or a NULL result. However, the successful identification of the prefix-suffix sequence triggers off the ASSIGNment of the appropriate property values for NGP, MODE, CASE GOVERNMENT, MOOD GOVERNMENT, CASE MARK, DEFINITENESS, REFERENCE, HUMANITY, and CATEGORY from the suffix to the Complex PF. The result RETURNed is a list of the inputword for CPFs, a list of the prefix and suffix for PGPs or the result of GENITIVIZE1. Thus, like GENITIVIZE1, FILTER5 is a procedure that

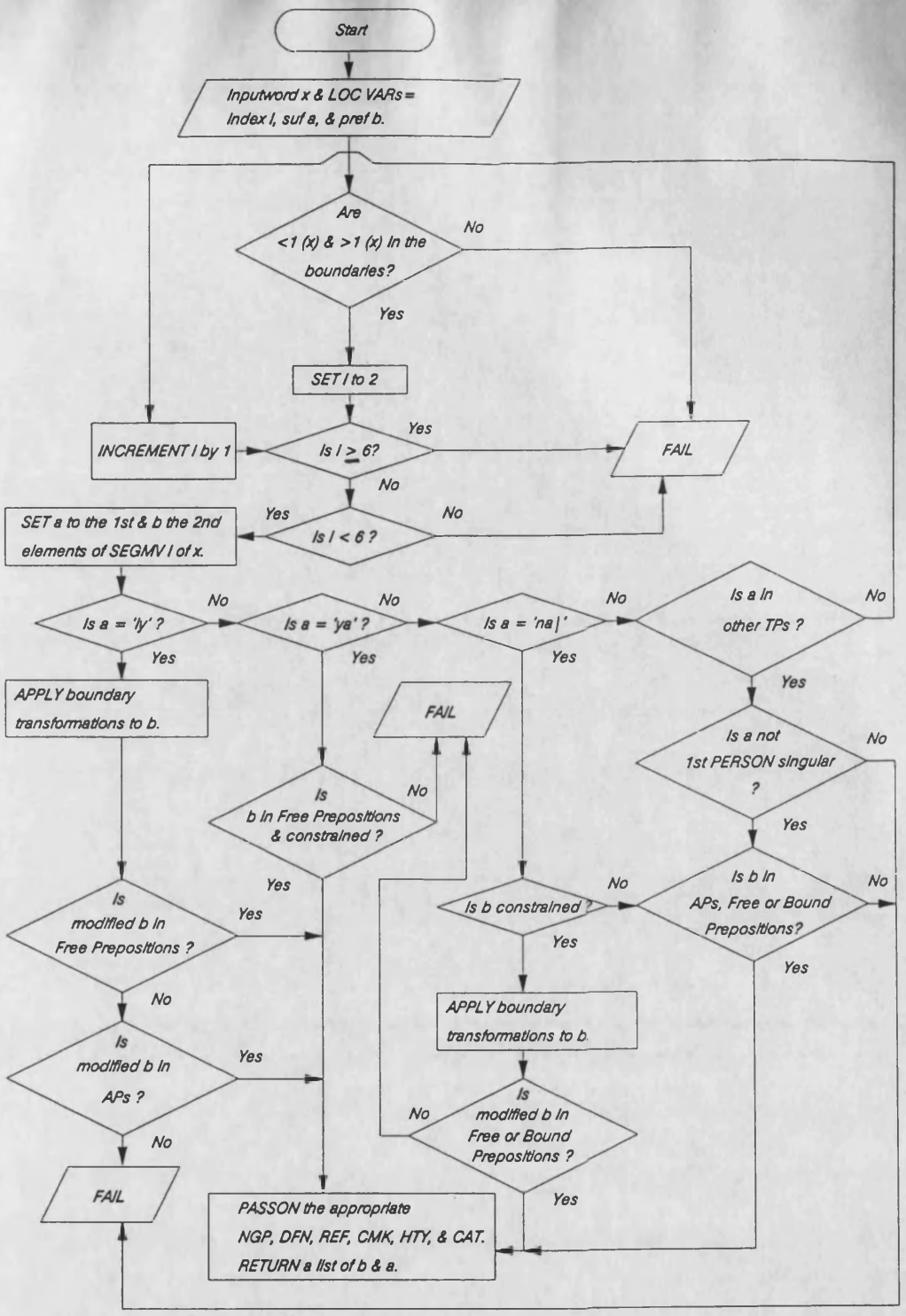


FIGURE 39: A Representation of the CPF & PGP Recognition Procedure

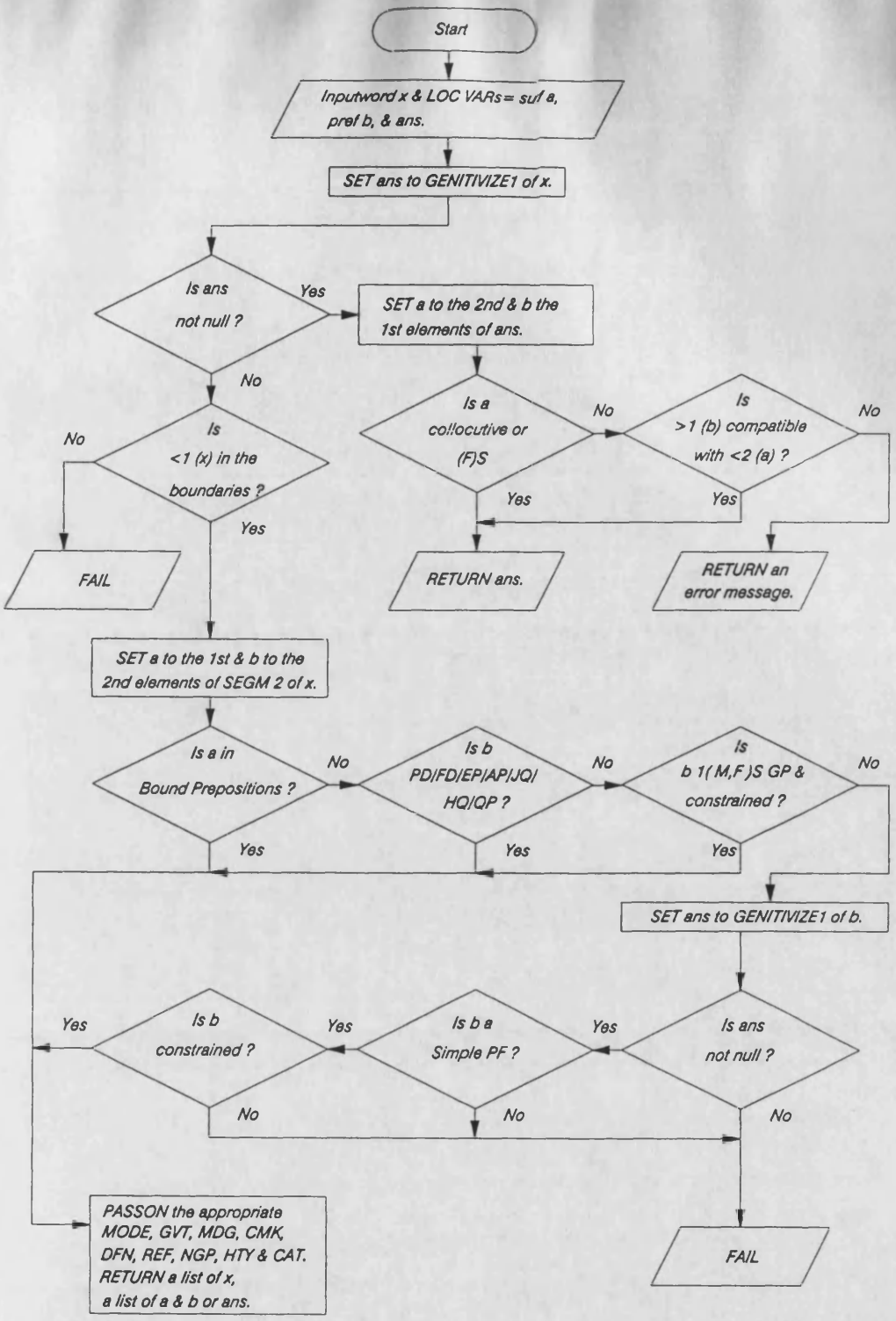


FIGURE 40:
A Representation of the Graphotactic Filtering & CPF & PGP Recognition Procedure

achieves recognition mainly through dictionary look-up but it also invokes GENITIVIZE1 as a lower procedure in order to process part of the input that it cannot identify. Figure 40 above represents FILTER5.

II SYNTHESIS OF THE P-COMPONENT

II.1 THE LOGICAL STRUCTURE OF THE P-COMPONENT

II.1.1 The Structure of the P-Complex

In Chapter 8, § II.1, we presented a typology of the Particle in M.S.A. and listed the various Particles selected for study with a formal account for them in Chapter 8, § II.2. The typology allowed us to classify the Particles into various kinds according to their syntactic function and their graphological and morphological structure. The formal account gave us a specification for graphotactic affixation rules governing the boundaries of Simple and Complex Particle Forms. We specified, for each of these PFs, a load of property values for features including NGP, DEFINITENESS, REFERENCE, CASE MARK, HUMANITY, CASE GOVERNMENT, MOOD GOVERNMENT, MODE, and CATEGORY. Earlier in this Chapter, we outlined LISP facilities for data representation, access, and property ASSIGNment in the P-Component. It also spelled out the recognition procedures for the data structures described. The formulation procedures were based on formal specifications for SPFs, CPFs, and PGPs. These specifications also included LENGTH, boundary, and lexical conditions and Annotated MS rules as described in Chapter 8.

Figure 41 below shows a general model for the P-Complex in M.S.A.

II.1.2 The Structure of the P-Processor

The P-Processor contains only two modules: GENITIVIZE1 and FILTER5. These two modules implement the linguistic descriptions of PGPs, SPFs, and CPFs as given in Chapter 8. Like the V- and N- Processors, the P-Processor implements boundary specifications as entry conditions, and LENGTH specifications as numerical indexes for alternative SEGMENTation and iteration. Affixation conditions are applied in the context-sensitive procedure FILTER5 which is invoked to enforce their observation in the output of GENITIVIZE1. CASE GOVERNMENT and Graphotactic Conditions are applied in both procedures as direct constraints on the progress of the parse. Allomorphic variation is handled through the use of simple transformations in order

	1	2	3
C P F	(BOUND PREPOSITION) L = 2	PARTICLE $3 \leq L < \infty$	(CLITIC PRONOUN) L = [2, 5]
P G P	BOUND PREPOSITION L = 2	Ø	CLITIC PRONOUN L = [2, 5]
C P F	Ø	PARTICLE $3 \leq L < \infty$	Ø
		..... *SIMPLE PF/PGP*	
..... *COMPLEX PF/PGP*			

FIGURE 41: A General Model for the P-Complex in M.S.A.

to simulate a MATCH between the inputword and a particular database form. Unlike the V- and N- Processors, the P-Processor does not involve any DERIVation, and after SEGMENTation and possible simple transformations, processing is carried out through dictionary look-up. The identification of MS-rule contexts triggers off the ASSIGNment of the appropriate feature values to the PF in question.

Processing in the P-Processor proceeds according to this logical framework:

<SEGMENTation>

<**case**: Bound-Preposition identification>

<**subcase**: Pronoun identification>/<**subcase**: invoke GENITIVIZE1 on suffix>

<SEGMENTation>

<**case**: Pronoun identification> <**subcase**: Pr/AP identification>

<**action**: constraint application, property ASSIGNment>

<**result**: output structure/error> <**repeat** SEGMENTation>.

II.1.3 The Architecture of the P-Processor

The P-Processor is a subprogram made up of two structured modules. FILTER5 invokes GENITIVIZE1 which resorts to the basic SEGMENTation subroutine to achieve decomposition of an input PF. There are no intermediate routines for DERIVation but boundary specifications are used as constraints on P-Database access and subsequently structures are identified by dictionary look-up. Both FILTER5 and GENITIVIZE1 carry out recognition procedures, the latter of Simple PFs and PGPs, and the former of Complex PFs and PGPs. In addition, FILTER5 performs a filtering task to validate or invalidate the analysis obtained by GENITIVIZE1 and

uses the Affix Selection Rules including compatibility conditions as the criteria for grammaticality. Both procedures make use of a number of LOCAL VARIABLES and one input argument to achieve one or more objectives and RETURN an error message, a NULL or a positive result.

In parallel to the other processors, the P-Processor also has a modular system architecture which is shown in Figure 42 below.

II.2 THE SEARCH PHILOSOPHY OF THE P-PROCESSOR

II.2.1 Converting M-Trees to P-Goal Trees

In Chapter 8, § II.2, we provided Annotated M-Markers for the rewriting of CPFs and PGPs. The implementation of these markers as an optimum-recognition process requires the conversion of such markers into P-Goal Trees. We have already defined (in Ch. 2, § III.1) how to convert an M-Tree into a G-Tree, what a G-Tree is and how to solve it. Figure 43 below shows a G-Tree for a Particle Complex and illustrates the ordered search paths for the search strategy of the P-Processor.

II.2.2 The Search Strategy and the Flow of P-Parsing

Figure 43 below shows a G-Tree for a Particle Complex with *or* and *and* nodes. So what is the optimum path that can be followed for the solution of this tree? Here again, the guiding principles in finding the answer are: 1) to aim for the simplest or minimal concatenations first, and 2) to proceed right-to-left first.

Thus, the P-Processor searches for a GP, and then looks for an obligatory Free or Bound Preposition. The next choice is to search right-to-left for a CP, and then look for an obligatory AP. If that FAILS, then the next choice is a left-to-right search for a Bound Preposition, and then an obligatory Free Particle (FP) which can be a JQ, HQ, QP, EP, PD or FD. If this FAILS, then the last choice is to look left-to-right for a Bound Preposition, and then right-to-left for an obligatory CP preceded by an obligatory AP that is prefixed to it. The successful identification of the above structural elements is accompanied by the application of CASE conditions and ASRs as constraints on the satisfaction of the nodes of the G-Tree. For each TP identification, an iteration is performed for a LENGTH interval: [2, 5] at the right of the inputword in GENITIVIZE1; and, for each Bound Preposition, a constant LENGTH requirement [2] is applied at the left of the inputword in FILTER5. First, GENITIVIZE1 is invoked on the whole of inputword, then FILTER5 is invoked to parse a prefix, and then GENITIVIZE1 is invoked again, but this time on the suffix. Thus, each exit from GENITIVIZE1 is controlled by FILTER5 in

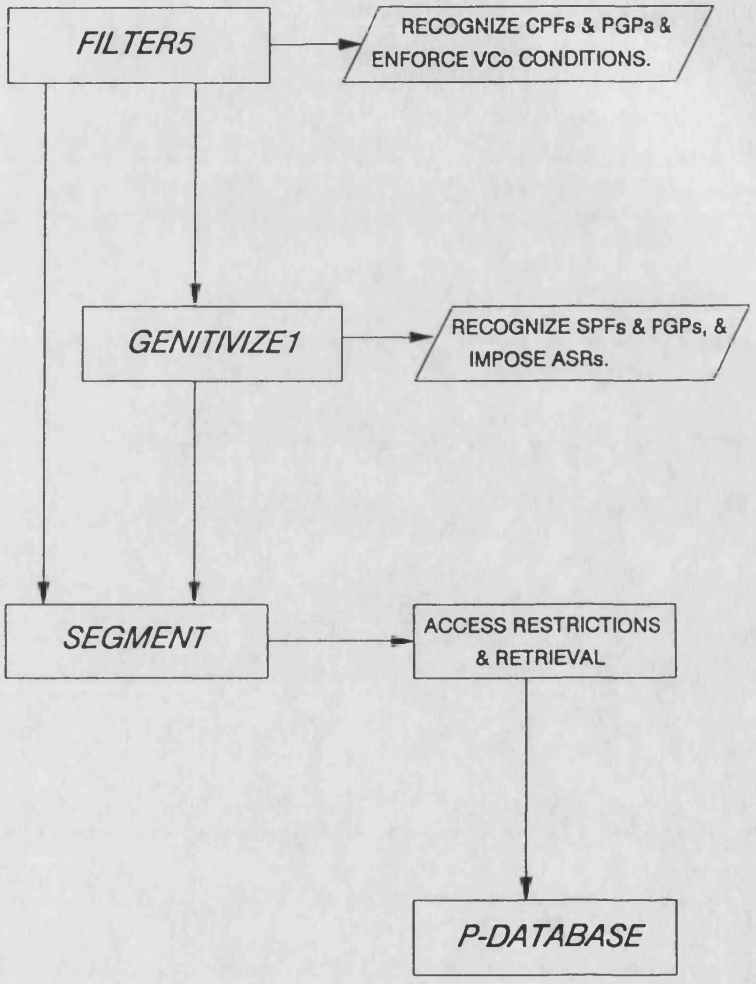


FIGURE 42: The Architecture of the P-Processor in TUNIS1

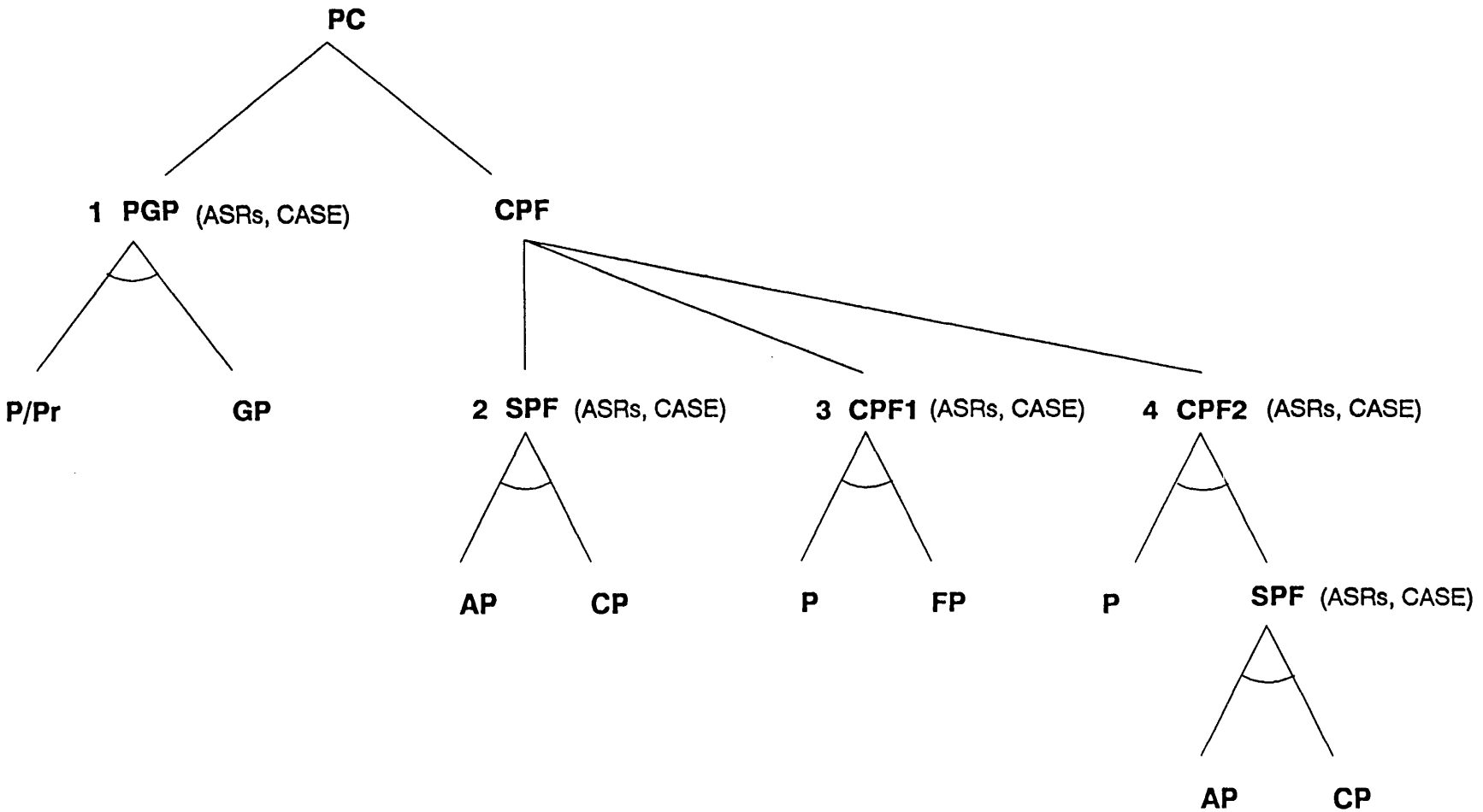


FIGURE 43: Search Paths and Constraints in a G-Tree for the P-Complex in M.S.A.

the requirement of CASE constraints and Graphotactic Conditions of compatibility in its output. Figure 44 below illustrates the progress and direction of parsing in the P-Processor.

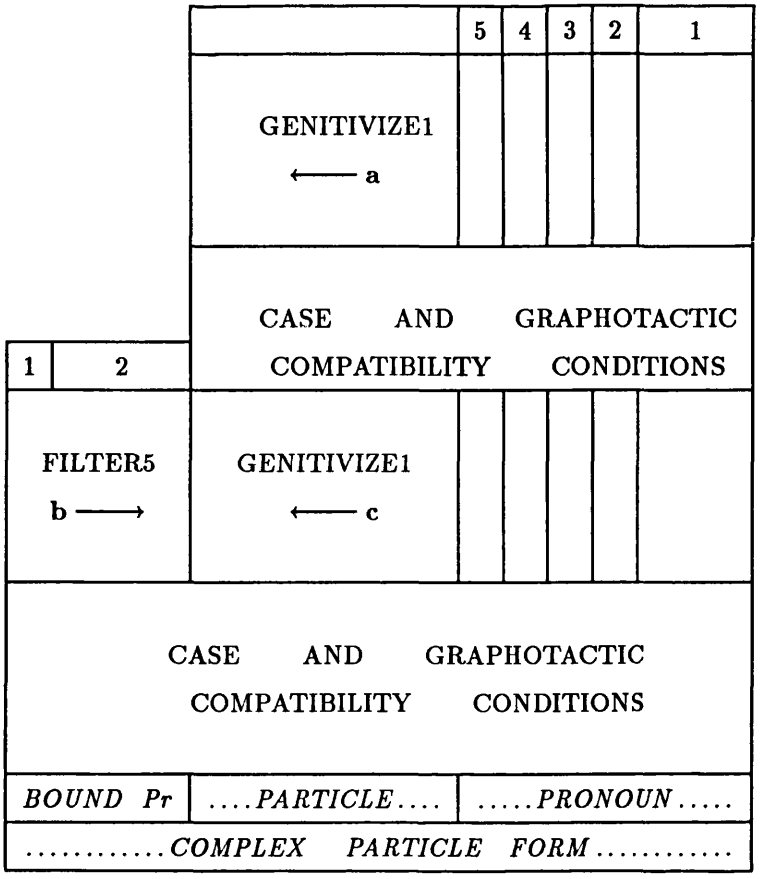


FIGURE 44: A Chart for the Direction of Parsing in the P-Processor

Like the V- and N- Processor, the P-Processor also implements a depth-first exhaustive search without backtracking. We have previously argued that this technique is dictated by the linguistic principles described and by the need for context-sensitive property ASSIGNment and structural and property disambiguation. We have also argued that such factors as the above exclude context-free or blind search methods. Rather, the search technique in the P-Processor is constrained with the application of the linguistic rules which are implemented as exit conditions on node satisfaction, which leads to a reduction in search space and therefore parsing time.

Note that, here also, the P-Processor flattens the M-Trees it generates. From Figure 41, we can deduce possible bracketed structures of the form: [(P) [AP/Pr (TP)]]. However, only single or unified categorial descriptors are actually generated, such as: PGP, PQP, PHQ, PJQ, PPD, PEP, and PFD—for Simple PFs—and PPC, for Complex PFs. For further illustration of the procedures of the P-Processor, we provide some sample morphological parses by TUNIS1 where the function MKTRUNK invokes FILTER5 as a subprogram (cf. Vol. II, App. D, § A).

==0==<*****>==0==

Part V

THE POLYSYNTHETIC WORD IN M.S.A.

Chapter 10

AN INTEGRATED FORMAL MODEL OF THE POLYSYNTHETIC WORD

INTRODUCTION

We have so far analysed the morphological structure of Complexes such as the Verb, Nominal or Particle ones. However, we need to provide an integrated formal model for the structure of any Arabic morpheme in general. Since a morphological processor has to be able to analyse a given inputword, without being told what type of Complex it is, we have to provide a typology for complex words in M.S.A. and for the type of antefix that can be attached to them, as well as a study of all the different types of morphological CATEGORIES that can be generated.

Such a study should yield a generalized model for *Polysynthetic* structures in M.S.A., which we define below, and by which we can account for all Complexes including esoteric classes such as *Time* and *Place Adverbs*. The study of complex-morpheme structure should also provide specifications for affixation conditions inside *Polysynthetic* structures, for disambiguation ensuing from such affixation, and for adequate Annotated MS rules that can generate "all and only" valid *Polysynthetic* morphological sequences. The above results, if obtained, should make possible the expression of an efficient parser for such sequences.

I A LINGUISTIC DESCRIPTION OF THE POLYSYNTHETIC WORD IN M.S.A.

I.1 A TYPOLOGY OF THE POLYSYNTHETIC WORD IN M.S.A.

I.1.1 Antefixes and Classes of Polysynthetic Structure

In Chapter 8, § I.1.2, and in Table 54, we have already covered the Question Particle and the Coordinative Conjunctions. The Question Particle, or Q, ':a', which corresponds to the French: 'est-ce que?', loosely, "is it true that?", and the Coordinative Particles 'wa, fa', "and" are all Bound Particles. They are also the elements which occupy antefix position in Arabic Morphological structures. We shall call such structures *Polysynthetic Words* or morphemes, which we define as structures containing a maximum grammatical number of slots where morphological formatives, including antefixes and major (i.e., V-, N-, or P-) Complexes, can be filled in (thus, our use of the term *Polysynthetic* differs from its orthodox use to denote languages with no free morphemes). Both Q and C are optional antefixes, Q always precedes C, and both can be antefixed to any of the major Complexes investigated so far. Therefore, there are four types of Polysynthetic Words in M.S.A. as follows:

a. A Verb Polysynthetic Word, i.e, QCV-Complex:

1	2	3	4	5	6	7
Q	C	F	P	Ce	S	CP
':a	fa	sa	t	udarris	uwna	humo?'
"Q-	and-	will-	you-	teach-	2 (M) P	them 3 (M) P?"
"	and	will	you	2 (M) P	teach	them 3 (M) P?

b. A Nominal Polysynthetic Word, i.e., QCN-Complex:

b.i. Referential:

1	2	3	4	5	6	7
Q	C	P	Ce	G	K	GP
':a	wa	bi	madoras	at	ayo	himo?'
"Q-	and-	in-	schools-	(F)	D	their 3 (M) P?"
"	and	is there in	their	3 (M) P	two	schools 3 (F) D?"

b.ii. Definite:

1	2	3	4	5	6	7
Q	C	P	D	Ce	G	K
':a	wa	bi	:alo	madoras	at	ayoni?'
"Q-	and-	in-	the-	schools-	(F)	D?"
"	and	is there in	the	two	schools 3 (F)	D?"

c. A Particle Polysynthetic Word, i.e., QCP-Complex:

c.i. Affirmative:

1	2	3	4	5
Q	C	P	AP	CP
':a	wa	li	:anna	ha|?'
"Q-	and-	because-	indeed-	she 3 (F) S?"
"	and is it	indeed	because	she 3 (F) S?"

c.ii. Prepositional:

1	2	3	4
Q	C	Pr	GP
':a	wa	ɛinoda	hunna?'
"Q-	and-	with-	them 3 (F) P?"
"	and	do they 3 (F) P	have?"

d. A Free Particle:

1	2	3
Q	C	FP
':a	wa	huna|ka?'
"Q-	and-	there?"
"	and is	there?"

I.1.2 Morphological CATEGORIES in Polysynthetic Structure

The possible combinations inside Polysynthetic template structures make up the set of all the morphological CATEGORIES generated by TUNIS1, which are of four broad types as above. Each categorial symbol is either a simple one or a composite of two or more and which can serve as simple input to syntactic analysis. We have referred to the unification of multiple symbols into one as *tree flattening*. We account for all the possible grammatical and ungrammatical categorial combinations in TUNIS1 (of which we give examples in Vol. II (App. A, § C) in Tables 57–59 below, where P is the set: {li, la, bi, ka, sa}, C is the set: {wa, fa}, and Q is ':a'.

PREF/ VC-NC	1 Ø	2 P	3 C	4 Q	5 CP	6 QP	7 QC	8 QCP
Ø	Ø	*	*	*	*	*	*	*
VB	VB	PVB	CVB	QVB	CPVB	QPVB	QCVB	QCPVB
VR	VR	PVR	CVR	QVR	CPVR	QPVR	QCVR	QCPVR
VN	VN	*PVN	CVN	QVN	*CPVN	*QPVN	QCVN	*QCPVN
VA	VA	*PVA	CVA	*QVA	*CPVA	*QPVA	*QCVA	*QCPVA
VM	VM	*PVM	CVM	*QVM	*CPVM	*QPVM	*QCVM	*QCPVM
WV	WV	*PWV	CWV	*QWV	*CPWV	*QPWV	*QCWV	*QCPWV
NN	NN	PNN	CNN	QNN	CPNN	QPNN	QCNN	QCPNN
DN	DN	PDN	CDN	QDN	CPDN	QPDN	QCDN	QCPDN
MM	MM	PMM	CMM	QMM	CPMM	QPMM	QCMM	QCPMM
DM	DM	PDM	CDM	QDM	CPDM	QPDM	QCDM	QCPDM
AA	AA	PAA	CAA	QAA	CPAA	QPAA	QCAA	QCPAA
DA	DA	PDA	CDA	QDA	CPDA	QPDA	QCDA	QCPDA
L1	L1	PL1	CL1	QL1	CPL1	QPL1	QCL1	QCPL1
L2	L2	PL2	CL2	QL2	CPL2	QPL2	QCL2	QCPL2
MD	MD	PMD	CMD	QMD	CPMD	QPMD	QCMD	QCPMD
AN	AN	PAN	CAN	QAN	CPAN	QPAN	QCAN	QCPAN
XN	XN	PXN	CXN	QXN	CPXN	QPXN	QCXN	QCPXN
XA	XA	*PXA	CXA	*QXA	*CPXA	*QPXA	*QCXA	*QCPXA
WN	WN	PWN	CWN	QWN	CPWN	QPWN	QCWN	QCPWN

TABLE 57: Verb and Nominal PW-CATEGORIES

a. Prepositional:				
PREF/	1	2	3	4
SUF	Pr	CPr	QPr	QCPr
Ø	Pr	CPr	QPr	QCPr
GP	PGP	CPGP	QPGP	QCPGP

PREF/	1	2	3	4	5	6	7	8
SUF	Ø	P	C	Q	CP	QP	QC	QCP
Ø	Ø	*	*	*	*	*	*	*
GP	*	PGP	*CGP	*QGP	CPGP	QPGP	*QCGP	QCPGP

b. Affirmative:								
PREF/	1	2	3	4	5	6	7	8
SUF	AP	CAP	QAP	QCAP	PAP	CPAP	QPAP	QCPAP
Ø	AP	CAP	QAP	QCAP	PP	CPP	QPP	QCPP
CP	PC	CPC	QPC	QCPC	PPC	CPPC	QPPC	QCPPC

TABLE 58: Prepositional and Affirmative PW-CATEGORIES

PREF/	1	2	3	4	5	6	7	8
FP	Ø	P	C	Q	CP	QP	QC	QCP
Ø	Ø	*	*	*	*	*	*	*
JP	JP	*PJP	CJP	QJP	*CPJP	*QPJP	QCJP	*QCPJP
FD	FD	PFD	CFD	QFD	CPFD	QPFD	QCFD	QCPFD
PD	PD	PPD	CPD	QPD	CPPD	QPPD	QCPD	QCPPD
SP	SP	*PSP	CSP	QSP	*CPSP	*QPSP	QCSP	*QCPSP
LD	LD	*PLD	CLD	QLD	*CPLD	*QPLD	QCLD	*QCPLD
AD	AD	*PAD	CAD	QAD	*CPAD	*QPAD	QCAD	*QCPAD
Q2	Q2	*PQ2	CQ2	*QQ2	*CPQ2	*QPQ2	*QCQ2	*QCPQ2
JQ	JQ	PJQ	CJQ	*QCJQ	*QJQ	*CPJQ	*QPJQ	*QCPJQ
EP	EP	*PEP	CEP	QCEP	QEP	*CPEP	*QPEP	*QCPEP
HQ	HQ	PHQ	CHQ	*QCHQ	*QHQ	*CPHQ	*QPHQ	*QCPHQ
QP	QP	PQP	CQP	*QCQP	*QQP	*CPQP	*QPQP	*QCPQP

TABLE 59: Free Particle PW-CATEGORIES

I.2 SCOPE OF THE MORPHOLOGICAL ANALYSIS IN TUNIS1

I.2.1 Esoteric Classes of Word

We have divided our analysis so far into three different parts: Verb, Nominal, and Particle. In each of these parts, we have included various word classes of Arabic. One CATEGORY of word that we have not dealt with so far is that of *Adverbs*. We can divide this CATEGORY into two types:

1) Place Adverbs, or LDs, such as 'huna|', "here", and 'huna|ka, huna|lika-', "there"; and

2) Time Adverbs, or ADs, such as ':alo:a|na', "now", 'baʕodu', "yet", and ':abada+', "never".

These Adverbs, like Particles, have invariable *primitive*, or non-derived, morphological forms but, unlike Particles, their meaning and functions are not dependent on other elements. They can, in fact, occur as main constituents in a sentence such as 'huna| tuwnis', "here is Tunis". Moreover, they can occur freely in Arabic sentences. Nevertheless, Place Adverbs also carry a DEFINITENESS value since they have exophoric REFERENCE. This DEFINITENESS value can serve as a condition for the positioning of Adverbs in initial sentence structures during syntactic parsing.

However, Adverbs are different from V-, N-, and P- Complexes in that they do not attach with any affixes except the Bound Interrogative and Coordinative antefixes. Therefore, their morphological and computational analysis had to be placed at a different point in the parsing process.

I.2.2 The Lexicon

We can now summarize the contents of the overall lexicon in TUNIS1. This is in fact a superset of CATEGORIES containing all the elements from each class of formative and amounting to 522 entries. These are divided up as follows:

a. 149 Verb and Nominal roots in rootlist.

b. 185 patterns including active and passive, perfect and imperfect Verb patterns, active and passive Verbal, Substantive, Tlocative, Adjective, and Numeral patterns.

c. Non-dynamic, or invariable, morphemes including Proper Nouns, Subject and Demonstrative Pronouns, and Time and Place Adverbs.

d. The list of Verb and Nominal Bound Particles and affixes including imperfect Verb prefixes and suffixes, perfect Verb suffixes, Nominal NGC suffixes, Bound Future Particle, Nominal Determiner, Bound Prepositions, Coordinative Conjunctions, Bound Interrogative Particle, and clitic Pronouns.

e. The list of Free Verb and Nominal Particles including Free Prepositions, Affirmative, Negative, Jussive, Subjunctive, Future, Assertive, and Interrogative Particles.

f. Finally, there is a list of the graphemes of Arabic, including consonants, vowels, and semivowels. These are used by the root and stem EXTRACTion routines which have to distinguish between such types of grapheme in order to find the root or stem.

There is a small degree of redundancy in the lexical entries. For instance, both Particles and affixes have different entries for allomorphic variants, and the dual suffix 'a|ni' is both in Verb and Nominal suffixes. However, as we have argued before, this redundancy in the lexicon allows us to avoid the use of complex generative transformations, and therefore, we also avoid redundancy and inefficiency in the Processor. The overall lexicon for TUNIS1 is provided in Volume II, Appendix B.

II A FORMAL ACCOUNT OF THE POLYSYNTHETIC WORD IN M.S.A.

II.1 SEQUENTIAL CONDITIONS OF AFFIXATION IN PWs

We have already distinguished Bound and Free Particles. Certain Free Particles are further distinguished by not accepting affixation to prefixes and suffixes. These Particles include JP, SP, FD, PD, and Q2. Adverbs are also subject to this constraint. Hence:

*P$Adv, and *P$FP where ADV $\longrightarrow$ LD/AD, and FP $\longrightarrow$ JP/SP/FD/PD/Q2.

Free Particles including JQ, EP, HQ, and QP can be affixed to certain bound prefixes or suffixes, subject to certain Graphotactic Conditions and ASRs as outlined in Chapter 8, § II.1.3. There are in fact constraints which ensue from these conditions, and which govern the affixation of the Interrogative and Coordinative Bound Particles to other Complexes. These constraints can be expressed as follows:

a. *P$JQ, and *P$HQ where P = 'la', "to". This implies also: *CPJQ, and *CPHQ under the same condition.

b. *P$QP where P = 'la', "to" and QP $\neq$ 'ma|d-a|', "what" implies also *CPQP under the same conditions.

c. *P$EP where P $\neq$ 'bi', "in, at ..." implies also *CPEP, and *QPEP under the same condition.

d. *P$GP where P $\neq$ {la, bi} ("to" and "in, at ...") implies also *CPGP, *QPGP, and *QCP$GP under the same condition.

e. *P$AP where P ≠ 'la', "to" and AP ≠ ':anna', "indeed" implies *PAPCP, *CPAP, *CPAP$CP, *Q$P$AP, *Q$PAPCP, *QCP$AP, and *Q$CPAP$CP under the same conditions.

We have also described general constraints governing Particle affixation to V- and N- Complexes such that:

f. *P$NC where P = 'la', "to" which implies *CPNC, *QPNC, and *QCP$NC under the same condition.

g. *P$VC where P ≠ {sa, li} ("going to", and "will") which implies *CPVC, *QPVC and *QCP$VC under the same condition.

There are also two specific constraints governing Q and C affixation in PWs. The first inhibits the affixation of Q, C or QC to all Free Particle variants where the variant is bound, such that:

h. *C$FP, *Q$FP, *QCFP where FP is a bound variant of a Free Particle. For example, ':ilayo' is the bound variant for ':ilay', "to, until". The allomorph ':ilay' can occur freely, but ':ilayo' can occur only when attached with a GP suffix. Hence, '*:awa:ilayo ...?', "*Q-and-to ...?", and '*:a$:ilayo ...?', "*Q-to ...?", are illegal.

The second constraint inhibits the affixation of Q to any structure with another interrogative element in it, such that:

i. *QX where X is interrogative. This implies the ungrammaticality of all the following sequences: *QPJQ, *QCP$JQ * Q$P$HQ, *Q$CPHQ, *QCQP, *QCP$QP. Hence, '*:a$ma|d-a|?', "*Q-what?", '*:a$mano?', "*Q-who?", and '*:awahalo?', "*Q-and-whether?", are all illegal. We shall refer to the constraint in *i.* as the *double-interrogative*, or *Q-Condition*.

We have already given some examples of ungrammatical CATEGORIES for the sequences in *a.* to *g.* In Tables 57–59 above, CATEGORIES which are inhibited, or disallowed, under all conditions are indicated with an asterisk. All other CATEGORIES are subject to the constraints detailed in Chapters 4, 6, 8, and 10.

II.2 CATEGORIAL DISAMBIGUATION ENSUING FROM PW-CONTEXTS

We notice that some of the CATEGORIES in Table 57 (rows 4–7 and 18–20) are homonymic or ambiguous CATEGORIES while others have unique categorial values. Upon the affixation of other elements to such homonymic CATEGORIES, we have already seen that some of the latter lose their ambiguity. For instance , 'd-ahaba', "gold/went" is an ambiguous VN for

Verb or Nominal, whereas 'd-ahaba$hu', "his gold" is an unambiguous Referential Nominall AN, since the Verb 'd-ahaba' requires a Prepositional not a Nominal Object. Similarly, we cam disambiguate other CATEGORIES upon affixation with Q and C elements as follows:

a. The CATEGORY XA which is an Abbreviated Nominal is ambiguous between Modifier and Adjective since it has an ambiguous FLEXION that does not indicate whether it precedes or follows the element it qualifies. Thus, ':uwlay', "first" can be a Modifier in ':uwlay :alobana|ti', "the first of the girls", or an Adjective: 'binotu+ :uwlay', "a first girl". However, if P or Q is affixed to such a Nominal, then XA can only be a Modifier, since both P and Q have priority to precede rather than follow other elements. We can formulate a rule such that:

XA $\longrightarrow$ MD / [ ] — P/Q/CP/QP/QC/QCP. Hence,

PXA $\longrightarrow$ PMD, QXA $\longrightarrow$ QMD, CPXA $\longrightarrow$ CPMD, etc.

b. The CATEGORY VM which is a perfect Verb or a Numeral in the accusative CASE becomes unambiguous after Q. This is because a Nominal is in the accusative only if it is preceded by another element, such as a Verb which governs it in that CASE, and since Q cannot be preceded by other elements. Thus, 'ɛasvara', "ten/to join ten ...", is a VN on its own, but, with the prefixation of Q, ':aɛasvara', can only be "did he join ten?", and not "!is-ten [+acm]?". We can deduce:

VM $\longrightarrow$ VB / [ ] — Q/QC.

c. Similarly, for the same reasons concerning GOVERNMENT as explained in *b.* above, the CATEGORY WV, for VR or AN, can be interpreted only as a VB when it follows a Q or QC. Thus, ':ah-asabahu' can be interpreted only as "did he count it?", and not "!is his honour?". Hence, we can say: WV $\longrightarrow$ VB / [ ] — Q/QC.

II.3 ANNOTATED MS RULES FOR THE PW

The specification, in Chapter 10, of sequential conditions of affixation and of CATEGORY contexts in fact implies a formal morphological grammar, or MS rules, for rewriting Polysynthetic Words in Arabic. Similarly to the grammars written for VCs, NCs, and PCs, we can augment such MS rules with annotations. However, there is only one requirement on an X-Complex, following Q, and that is for X not to have interrogative MODE.

We have already stated that an X-Complex, in M.S.A., can be a VC, a NC, a PC, an Adverb or an FP, for all of which we have provided respective CSGs in Chapters 4, 6, and 8. Thus, we are now in a position to provide a generalized Annotated Grammar for Polysynthetic words in M.S.A. Given W, for word, we can formulate the grammar in *a.* below:

a.i.	W	⟶	(Q [+MODE]) (C) X ([+MODE])
a.ii.	X	⟶	ADV/FP/PC/VC/NC
a.iii.	ADV	⟶	LD/AD
a.iv.	FP	⟶	JP/SP/FD/PD/Q2/JQ/EP/HQ/QP/AP/Pr
a.v.	PC	⟶	PGP/CPF
a.vi.	VC	⟶	FCP
a.vii.	NC	⟶	PNF
a.viii.	LD	⟶	{huna|/huna|ka/huna|lika}
a.ix.	AD	⟶	{:alo:a|na/baɛodu/:abada+}
a.x.	JP	⟶	'lamo'
a.xi.	SP	⟶	'lano'
a.xii.	Q2	⟶	'halo'

Grammar *a.* generates grammatical Polysynthetic sequences with two optional Interrogative Q and Coordinative C elements as antefixes. Rule *a.i.* allows the generation of a sequence Q (C) X $\mathcal{IFF}$ the MODE of X is the opposite of that of Q. Rules *a.ii.* to *a.vii.* are also non-terminal rules which summarize the rewriting grammars specified so far (in Chapters 4, 6, and 8). Rules *a.viii.* to *a.xii.* are lexical rules which specify the terminal CATEGORIES not specified so far. Using this grammar we can generate bracketed markers for PWs of the general type: [W ([Q . :a [+int]]) ([C . wa/fa]) [X (+MODE)]], where X is as specified in the grammars. However, the final result of the MA will be a flat single CATEGORY and not a multiple-node tree.

Using the grammar in *a.* we can generate non-Polysynthetic sequences, to wit, sequences without Q, and C elements as described in the grammars of Chapters 4, 6, and 8, above and we can also generate all the following types of PW-sequences:

b.i.	':a$satakotubu?', "will you write?":	Q$FCF
b.ii.	':afa$satakotubu?', "and will you write?":	QC$FCF
b.iii.	':afa$satakotubuhu?', "and will you write it?":	QC$FCF
b.iv.	':a$bi:alomanozili?', "is there in the house?":	Q$PNF
b.v.	':awa$bimakotabihi?', "and does his desk have?":	QC$PNF
b.vi.	':afa$lida|risi+?', "and does a student have?":	QC$PNF
b.vii.	':awa$huna|ka?', "and is there?":	QC$ADV
b.viii.	':awa$qabolahumo?', "and is there before them?":	QC$PGP
b.ix.	':awa$lamo ...?', "and won't ...?":	QC$FP
b.x.	'wa$halo ...?', "and whether ...?":	QC$FP

==0==<★★★★★>==0==

Chapter 11

PROCESSING THE POLYSYNTHETIC WORD

INTRODUCTION

Chapter 10 gives a formal account of Polysynthetic Words. In Chapter 11, we convert the definitions given in this account to formal specifications that spell out the requirements for LENGTH, boundary, and lexical conditions within PWs. We then implement these specifications as an efficient parser for PWs, which we will call the *PW-Processor*. Such a Processor has to find optimal methods and conditions for the access of the various databases built in the previous processors.

Further, Chapter 11 has to make clear the processes carried out at the top level for each procedure, describe the relationships of the various modules in the PW-Processor to each other, and explain how the linguistic rules of Chapter 10 are implemented as computational procedures in Chapter 11.

Finally, this Chapter has to synthesize the overall analysis for all the major Complexes and Processors of TUNIS1 by answering the questions on how to relate the linguistic to the computational, what kind of model we can build for PWs, what kind of architecture is designed for the PW-Processor, what kind of search strategy is adopted for PWs, what kind of G-Tree is used to represent MS rules for PWs, and finally how the morphological parsing proceeds in the general TUNIS1 system.

I CONTENTS OF THE PW-COMPONENT

I.1 THE PW-DATABASE

In this Chapter, we make parallel assumptions to those in Chapter 9, § I.1.1. Further, the PW-Database can be considered as a set of all the databases in TUNIS1 including the V-, N-, and P-Databases.

The PW-Database also makes use of lexical entries SET up in FWT, which is the dictionary for Free Particles whose contents are as detailed in Chapter 9, § I.1.2, *a.–d.* Each entry in FWT carries unique default values ASSIGNed for properties such as MOOD and CASE GOVERNMENT, and MODE. Further, entries in FWT are static or fixed with respect to their structure as opposed to dynamic entries such as roots and patterns.

I.2 THE PW-PROCESSOR

I.2.1 Initial Closed Subroutines

In order to handle TRANSCRIPTION, SEGMENTation, word ASSEMBLY, character PICKing, COPYing, SUBSTITUTion, and DELETion, the PW-Processor makes use of the apparatus of independently-defined closed subroutines as specified in Chapter 3.

I.2.2 The Main PW-Recognition Procedure: QUESTION

The formal account and descriptions of the Polysynthetic word in M.S.A., as given in Chapter 10, imply a formal specification for PWs such as in Table 60 below.

We can now define QUESTION: the main PW-Recognition procedure based on the specifications of Table 60. First, we define a function called *YPARSE* which invokes FILTER5, FUTURIZE, and GENITIVIZE2, in this order, and RETURNS the first non-NIL result that it gets from these functions. Given an inputword w as an argument, QUESTION can then work on this word by invoking YPARSE as a sub-process. The task of QUESTION is to recognize a PW that is an Adverb, a simple Free Particle (FP), or a number of segments which are optional Interrogative and Coordinative Particle prefixes followed by an Adverb, an FP or a major Complex: PC, VC or NC.

QUESTION starts by searching the Function Word Table: FWT for an Adverb or a simple FP. If it is successful it RETURNS the word, otherwise it invokes YPARSE which will call the

DESCRIPTION TYPE	an abstract composite template structure.
STRUCTURE	*BASIC STRUCTURE:* sequences of: optional Bound Interrogative and Coordinative Particle slots, followed by an X sub-template for ADV, FP, PC, VC, or NC. *CONCATENATION ORDER:* given as (Q) + (C) + X. *STRUCTURAL TYPES:* (a) *Interrogative*: Q\$$X$. (b) *Coordinative*: C\$$X$. (c) *Interrogative Coordinative*: Q\$C\$$X$.
CONSTRAINTS	*GRAPHOTACTIC CONDITIONS:* initial Boundary Conditions. *Q LENGTH:* L = [2]. *C LENGTH:* L = [2]. *PW-LENGTH:* (a) *if* Q then LENGTH (PW) > 2 PLUS LENGTH (X); and (b) *if* C then LENGTH (PW) > 2 PLUS LENGTH (X); and (c) *if* Q & C then LENGTH (PW) > 4 PLUS LENGTH (X); (d) else, LENGTH (PW) = LENGTH (X). *PW-BOUNDARIES:* (a) *if* Q then <1 (PW) = <1 (Q), and <1 (Q) = <:>; and (b) *if* C then <1 (PW) = <1 (C), and <1 (C) = {f/w}; (c) else <1 (PW) = <1 (X). *LEXICAL SPECIFICATION:* *if* Q then, MODE (X) $\neq$ interrogative.
ROLE	provide a template for the realization of PWs.

TABLE 60: A Formal Specification for Polysynthetic Words

major Complex processors each in turn, until a result is obtained. If these FAIL, then if the initial character of w is in the PW boundaries, then QUESTION proceeds in a left-to-right manner, SEGMENTing w into two parts: a prefix of LENGTH 2 and a centre. If this prefix is a Coordinative Conjunction, then the above process is repeated in order to identify the centre. If the prefix is a Bound Interrogative Particle (Q), then the same process is repeated to parse the centre, and if that process FAILS, then the centre itself is SEGMENTed into two parts, and if the first part is a Conjunction, then the said process is again repeated for the centre.

For each prefix in FWT, the prefix must not be a bound allomorphic variant and, for each

centre following Q, the centre must not be interrogative. Upon identification of the constituent parts of w, QUESTION ASSIGNS to it the appropriate values for CASE GOVERNMENT, MOOD GOVERNMENT, MODE, and CATEGORY, and transfers to it the PLIST of the centre. Then, QUESTION RETURNS a result which is w, for non-referential morphemes, and a list of the prefix-centre sequence and the clitic Pronoun, for referential morphemes. The procedural context-sensitive routine QUESTION is represented in Figure 45 below.

I.2.3 Categorial Filtering: FILTER6

In Chapter 10, § II.1 and § II.2, we outlined some idiosyncratic conditions for affixation in PWs, and for the generation of ungrammatical CATEGORIES in CPFs and PGPs; we also described specific contexts for disambiguation within major Complexes such as VC and NC. We have not so far implemented these esoteric constraints. The task of *FILTER6* is to do exactly that.

While inspecting the output of QUESTION, FILTER6 MODIFIes a given CATEGORY XA to MD, a CATEGORY VM to VB, and a CATEGORY WV to VB in the contexts specified in Chapter 10, § II.2. FILTER6 then rejects any of the sequences QPP, QPPC, QCPP, or QCPPC, whenever they contravene the conditions in Chapter 10, § II.1. The sequences *QPJQ, *QCPJQ, *QPHQ, *QCPHQ, *QPQP, and *QCPQP, which violate the double-interrogative constraint (referred to as the *Q-Condition* and described in Ch. 10, II.1.*i*.) are rejected outright. Otherwise, the result of QUESTION is RETURNed. If all this FAILS, then FILTER6 FAILS.

Thus, FILTER6 does not carry out any processing, rather it modifies the output of a lower procedure: QUESTION, through categorial context scrutiny.

I.2.4 Top-Level Control in TUNIS1

The procedures described up to FILTER6 handle one word at a time. However, we need top-level routines that invoke these procedures for a list of words, or a *sentence*.

I.2.4.1 The Sentence-Recognition Procedure

First, we define a procedural function called *WPARSE* which takes an inputword w for a PW and invokes FILTER6 to process it. WPARSE then inspects the output structure from FILTER6 and, if it is an error, RETURNS an error message, if it has no values for PERSON, ASSIGNS a default PERSON value third to it; otherwise, it RETURNS the result of FILTER6. If WPARSE FAILS to recognize a word, it RETURNS a general error message informing the user that the word is unknown.

We can now define a sentence-recognition procedure which will wheel off a stream of words to WPARSE for morphological processing. *MPARSE* is such a procedure which, given an

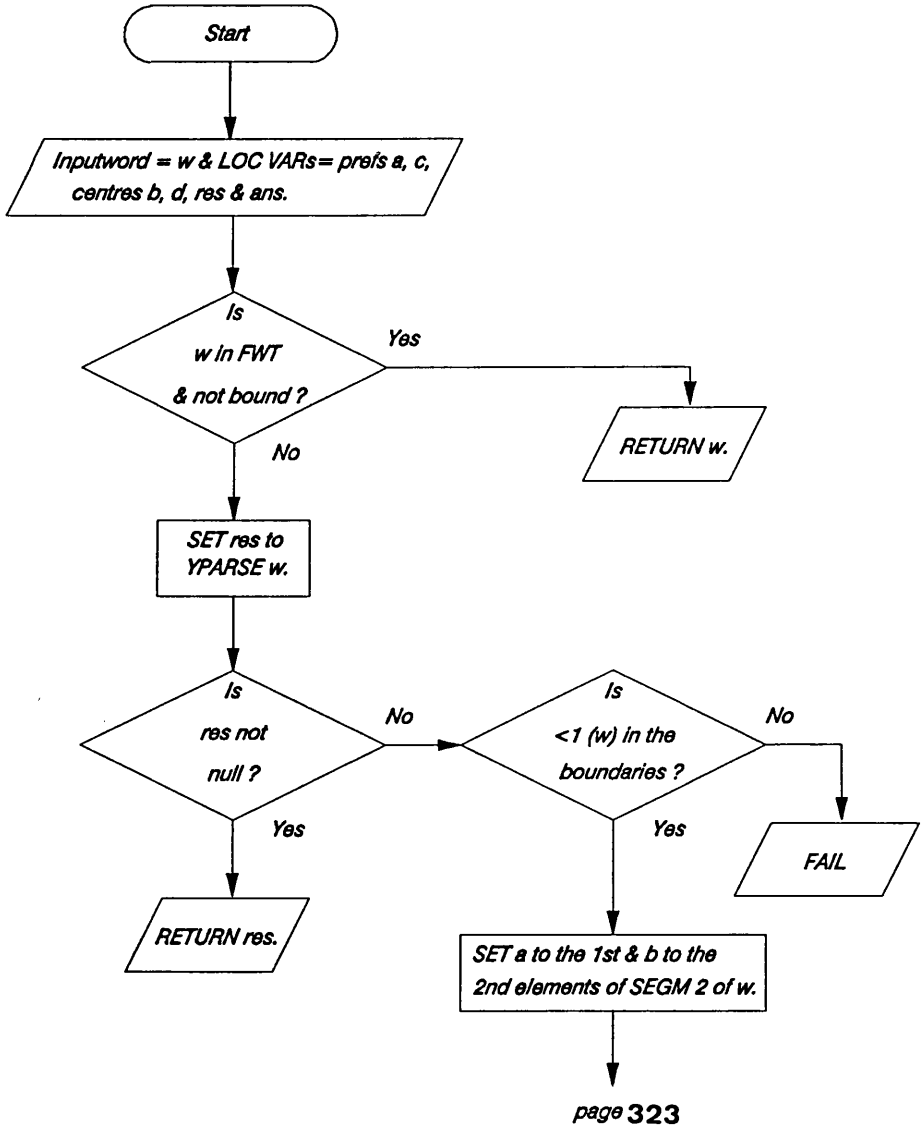


FIGURE 45: A Representation of the PW Recognition Procedure

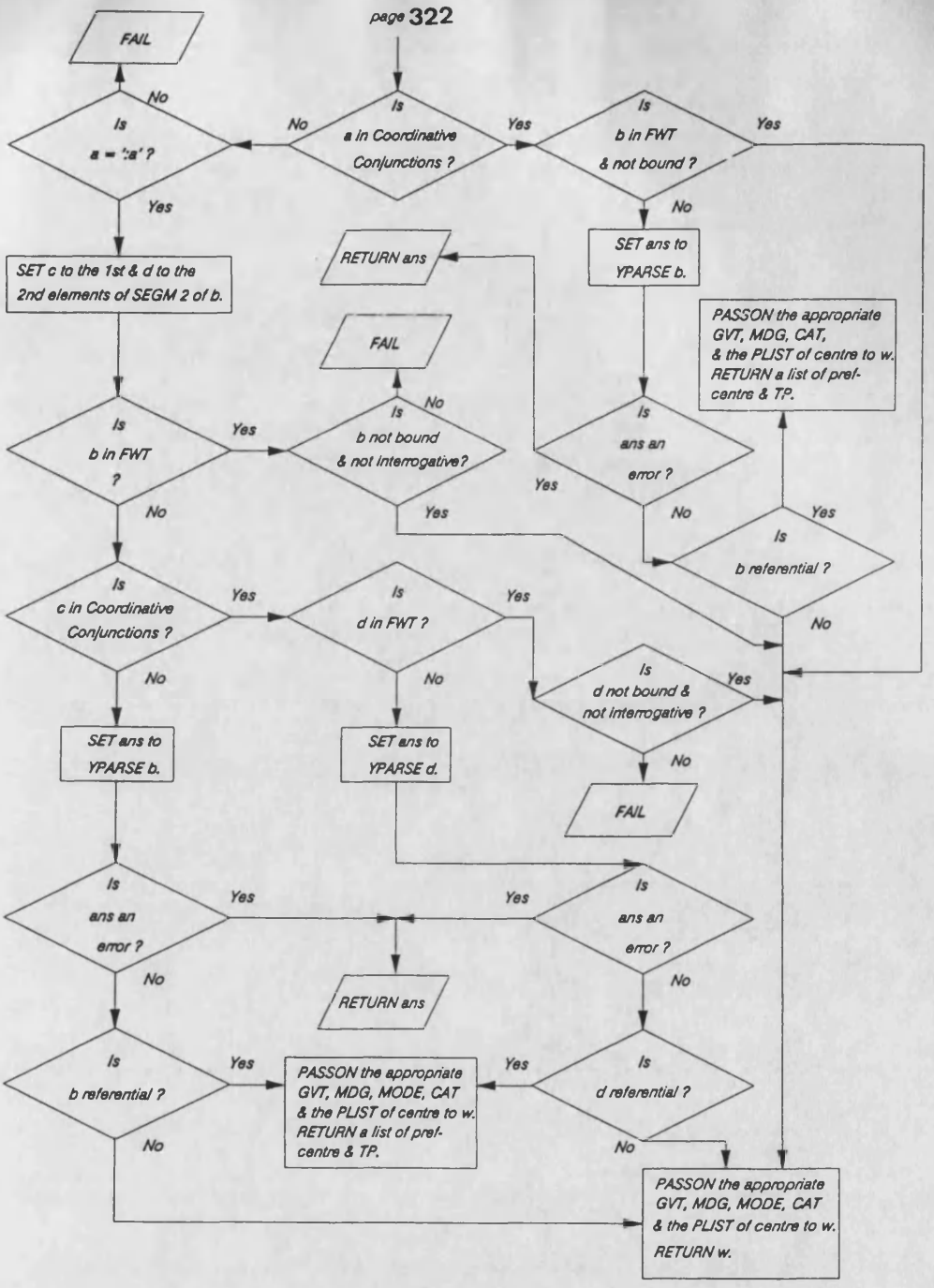


FIGURE 45: A Representation of the PW Recognition Procedure (Contd)

argument, which is a list of words, operates on it in a left-to-right direction, and performs an iteration for the LENGTH of the list.

In each traverse of the loop and for each output structure from WPARSE, MPARSE distinguishes referential morphemes, for which it places a COPY of the clitic Pronoun, within that structure, at a global register, or stack. This is because it is anticipated that this register will be useful in syntactic parsing for the implementation of the rules concerning REFERENCE and anaphora, in Arabic (cf. Ch. 12, § I.4.4). After this distinction, output structures are PUSHed into a temporary register, or stack, and when the words in the list are exhausted, then the REVERSE of the stack is RETURNed. The stack is REVERSed in order to preserve the same order as the sentence, given that a stack holds elements in a last-in first-out order (cf. Ch. 3, § I.3). If WPARSE RETURNS a FAILure, viz., an error or a NULL result at any point, then MPARSE also FAILS. This means that parsing is aborted if one word in the input list is not grammatical even if all the other words are, which means great savings in terms of syntactic parsing: syntactic analysis will be undertaken only after morphological parsing is successfully finished for all the sentence.

Note that, here, we mean by *sentence* merely a list of inputwords and the notion of grammaticality is restricted to morphological but not syntactic criteria. MPARSE is represented in Figure 46 below.

I.2.4.2 The M-Tree Generation Procedure

Having conveyed a stream of words through MPARSE and obtained a sequence of output structures from WPARSE for each word, we now need a general procedure to generate output primitive M-Trees, or M-Markers, that can be used as input structures to an eventual syntactic parser. We will call such a primitive marker a *TRUNK*.

First, we define a procedural function called *EXSYMBOLS* whose task it is to examine or extract CATEGORY symbols for each terminal, or lexical, item that is in each output structure in the sequence obtained from MPARSE. Given such a structure, EXSYMBOLS essentially checks whether it is referential or not and, if it is not, it creates a PAIR from the CATEGORY of the structure and the terminal symbol in it. If the structure is referential, then EXSYMBOLS creates a list of two PAIRS, one for the CATEGORY and terminal symbol of the structure, and one for the CATEGORY and lexical item of the clitic Pronoun which is in this structure.

We can now define a general procedure called *MKTRUNK* which, given a sentence, first invokes MPARSE to perform recognition tasks on it, and then, if it gets a positive result, and for each output structure from MPARSE, it repeatedly invokes EXSYMBOLS and PUSHes the result into a temporary stack called *trunk* until the output structure is exhausted; whereupon MKTRUNK finally RETURNS the REVERSE of trunk. Thus, like MPARSE, MKTRUNK

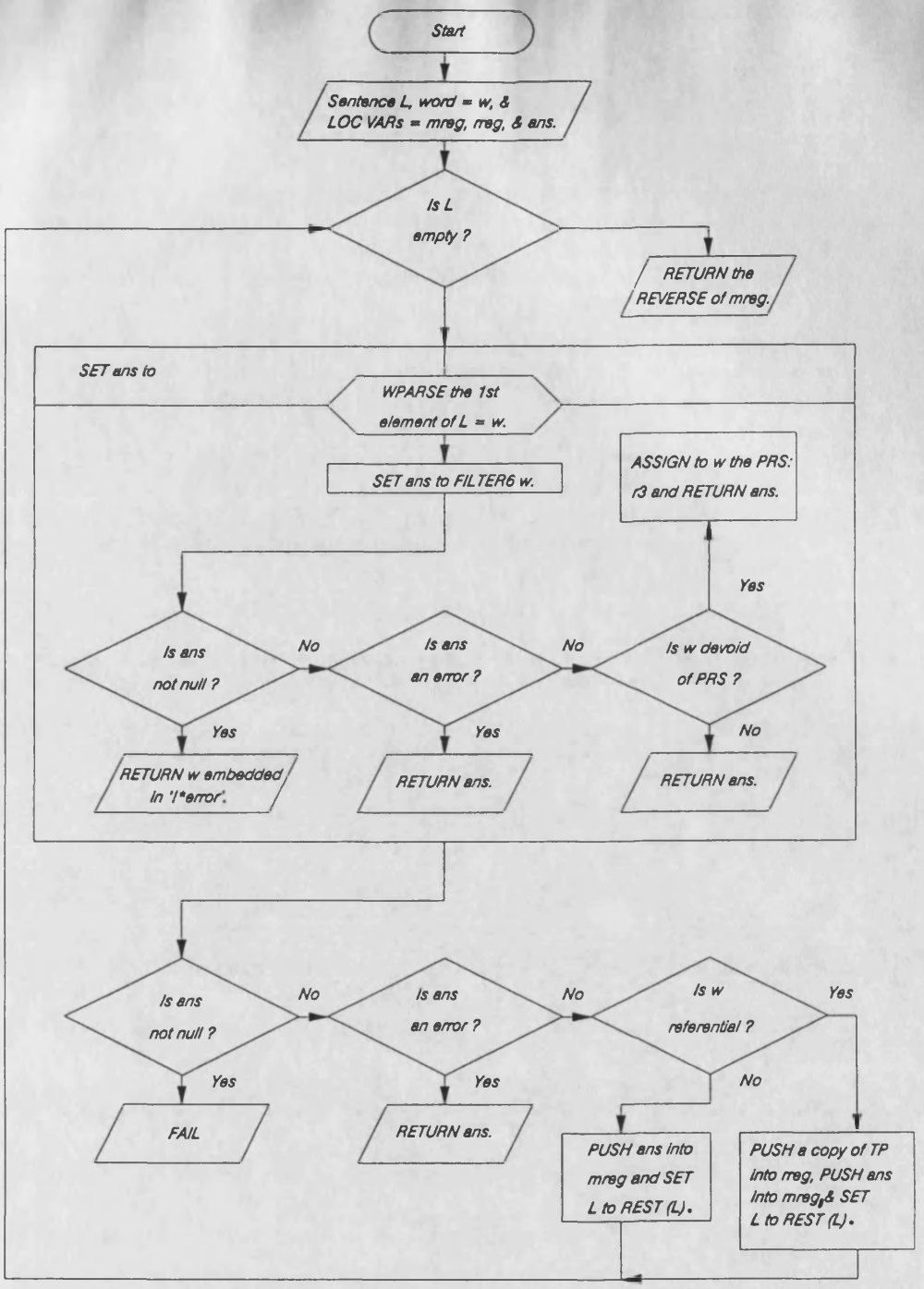


FIGURE 46: A Representation of the Sentence Recognition Procedure

is a left-to-right procedure which iterates for the LENGTH of a sentence. MKTRUNK is represented in Figure 47 below.

II SYNTHESIS OF THE PW-COMPONENT

II.1 THE LOGICAL STRUCTURE OF THE PW-COMPONENT

II.1.1 An Integrated Model for the Structure of PWs

In Chapter 10, § I.1, we described a typology of the Polysynthetic Word in M.S.A. and gave the various antefixes and classes of PWs, as well as the set of morphological CATEGORIES generated in TUNIS1. From these descriptions ensued a formal account for PWs with specifications for LENGTH, boundary, and lexical conditions and details of sequential conditions of affixation and categorial disambiguation in PWs. We argued that such an account implied formal Annotated MS rules for the generation of PWs and upon these Rules we based our general PW-Processor and its recognition procedures. Earlier, in Chapters 5, 7, and 9 we gave general models for the V-, N-, and P- Complexes respectively. Here is now an overall integrated model which encapsulates those earlier models in order to make up a general one for the Polysynthetic Word in Figure 48 below.

II.1.2 The General Structure of the PW-Processor

The PW-Processor is made up of five main modules. There are two top-level modules: MKTRUNK and MPARSE, which are context-free operations that handle the reading-in of sentences, or lists of inputwords, build M-Trees for them, and deal with referential structures. There are three BASIC modules: WPARSE which handles output structures from lower routines, and ASSIGNS default PERSON values to them, FILTER6 which implements categorial constraints and performs categorial disambiguation, and finally QUESTION which deals with the main tasks in the recognition of QCX, or PW sequences.

The PW-Processor in QUESTION, like other processors, implements boundary specifications as entry conditions and LENGTH specifications as numerical indexes for SEGMENTation. Affixation conditions such as the double-interrogative, or Q-Condition, are implemented directly in QUESTION.

The context-sensitive procedure: FILTER6 implements the idiosyncratic constraints observed in the generation of certain esoteric CATEGORIES in Chapter 10. It also performs some disambiguation for these CATEGORIES based on the analysis in Chapter 10, § II.2.

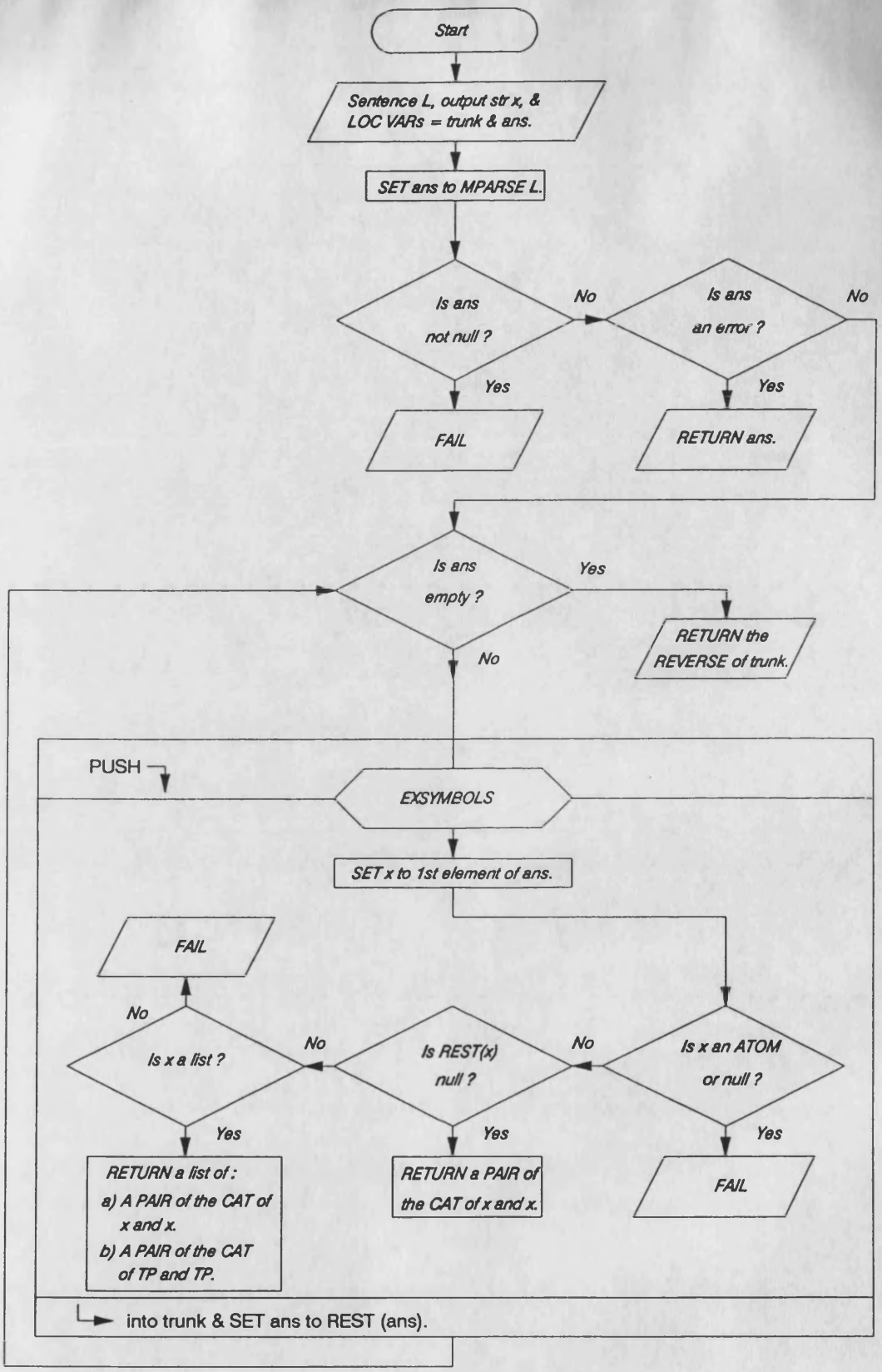


FIGURE 47: A Representation of the M-Tree Generation Procedure

<table>
<tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr>
<tr><td>Q</td><td>C</td><td>P</td><td>D</td><td>Ce</td><td>G</td><td>S/K</td><td>TP</td></tr>
<tr><td rowspan="6">(:a)
L= 2</td><td rowspan="6">(wa)
(fa)
L= 2</td><td>Ø</td><td>Ø</td><td>V-CENTRE
3 ≤ L < ∞</td><td>Ø</td><td>prf-
SUF
L= [1, 6]</td><td rowspan="3"></td></tr>
<tr><td>(sa)
(li)
L= 2</td><td>:/t/
y/n
L= 1</td><td>V-CENTRE
3 ≤ L < ∞</td><td>Ø</td><td>imp-
SUF
L= [1, 5]</td><td>(CLITIC
PRONOUN)
L= [2, 5]</td></tr>
<tr><td>(li)
(bi)
(ka)
L= 2</td><td>Ø</td><td rowspan="2">N-CENTRE
3 ≤ L < ∞</td><td rowspan="2">G-
SUF
[2, 3]</td><td rowspan="2">K-
SUF
[1, 5]</td><td></td></tr>
<tr><td>(:alo)
(:alC)
(lo) (lC)
L= [1, 4]</td><td>Ø</td></tr>
<tr><td>(li)
(bi)
(ka)
(la)
L= 2</td><td>Ø</td><td>PARTICLE
3 ≤ L < ∞</td><td>Ø</td><td>Ø</td><td>(CLITIC
PRONOUN)
L= [2, 5]</td></tr>
<tr><td>li
bi
ka
la
L= 2</td><td>Ø</td><td>Ø</td><td>Ø</td><td>Ø</td><td>CLITIC
PRONOUN
L= [2, 5]</td></tr>
<tr><td>Ø</td><td>Ø</td><td>PARTICLE/
ADVERB
3 ≤ L < ∞</td><td>Ø</td><td>Ø</td><td>Ø</td></tr>
<tr><td colspan="2">.....optional.....</td><td colspan="3">.....obligatory.....</td><td colspan="2">..optional..</td></tr>
<tr><td></td><td colspan="6">..... COORDINATIVE ..... POLYSYNTHETIC ..... WORD .....</td></tr>
<tr><td colspan="7">..... INTERROGATIVE ..... POLYSYNTHETIC ..... WORD .....</td></tr>
</table>

FIGURE 48: An Integrated General Model of PW-Structure in M.S.A.

At the appropriate points, the PW-Processor ASSIGNS to the inputword property values for CASE and MOOD GOVERNMENT, MODE, and CATEGORY as well as the dynamic PLIST resulting from invoking the major processors. The major Processors are invoked through

YPARSE which calls FILTER5 for PC processing and dictionary look-up, then FUTURIZE for VC processing, and then GENITIVIZE2 for NC processing. YPARSE RETURNS the first positive result it gets from any of these calls. QUESTION itself also performs dictionary look-up, for simple words, i.e., non-derived words which are listed in FWT, before it invokes YPARSE. Thus, processing in the PW-Processor is done in this overall logical framework:

<MKTRUNK (list)>
(EXSYMBOLS (output structures in list)>
<MPARSE (list)> <case: non-empty list>
<WPARSE word>
<FILTER6 word>
<QUESTION word>
<search word> <SEGMENTation>
<case: Conjunction context>
<search/process centre> <repeat SEGMENTation>
<case: Interrogative context>
<repeat search & process>
<subcase: Interrogative & Conjunction context>
<repeat search & process>
<action: property ASSIGNment>
<result: output structure>
<end QUESTION>
<case: categorial context>
<action: constraint application>
<result: output structure>
<end FILTER6>
<case: PERSON context>
<action: PERSON ASSIGNment>
<result: output structure>
<end WPARSE>
<case: referential context>
<action: PUSH output structures into stacks>
<repeat WPARSE> <result: stack contents>
<end MPARSE> <action: PUSH output structures into stacks>
<repeat EXSYMBOLS> <result: stack contents>
<end EXSYMBOLS>
<end MKTRUNK>

II.1.3 The Architecture of the PW-Processor

The PW-Processor is the main program in TUNIS1. It consists of five main modules being MKTRUNK, MPARSE, WPARSE, FILTER6, and QUESTION as described above. The basic procedure: QUESTION starts by searching for the whole inputword in FWT just in case it is an Adverb or an FP. If it is does not find either of these, QUESTION then attempts to parse the whole inputword and, if it FAILS, it SEGMENTS the word and looks for Interrogative and Coordinative antefixes. If it finds them, then it resumes dictionary look-up and then parsing. This parsing is done by invoking *YPARSE* which is a dummy function for traffic control. This traffic involves the coordination of output results from FILTER5, FUTURIZE, and GENITIVIZE2, whichever of them RETURNS a positive result first. A FAILure for all of them causes a general error message to be RETURNed by WPARSE. We have already made the point that MKTRUNK and WPARSE handle lists of words and M-Tree Generation, while lower modules handle one word at a time. The modular structure of the PW-Processor is shown in Figure 49 below, while the following Figures 50 and 51, show the overall logical structure of the Databases and the TUNIS1 morphological system.

II.2 THE SEARCH PHILOSOPHY OF THE PW-PROCESSOR

II.2.1 Converting M-Trees to PW-Goal Trees

Chapter 10, § II.3, provides Annotated MS rules for the generation of PWs. In order to achieve optimal recognition for the M-Trees derived from these rules, we can convert them into PW G-Trees. The definition and Conversion Rules for G-Trees have already been given in Chapter 2, § III.1. Figure 52 below shows a G-Tree with ordered search paths for the Polysynthetic Word.

II.2.2 The Search Strategy and the Flow of PW-Parsing

Figure 52, above, represents a set of obligatory *and* nodes and choice *or* nodes in a G-Tree for a PW. In order to achieve an efficient solution of this tree, we have to find an optimal route to the goal which is the solution. We have already related such decisions to two guiding and general principles and these are: 1) to look for minimal concatenations first, and 2) to proceed right-to-left.

The minimal-LENGTH structures here are in fact Adverbs and FPs that are devoid of affixes. Accordingly, the PW-Processor performs dictionary look-up for these first. If it FAILS,

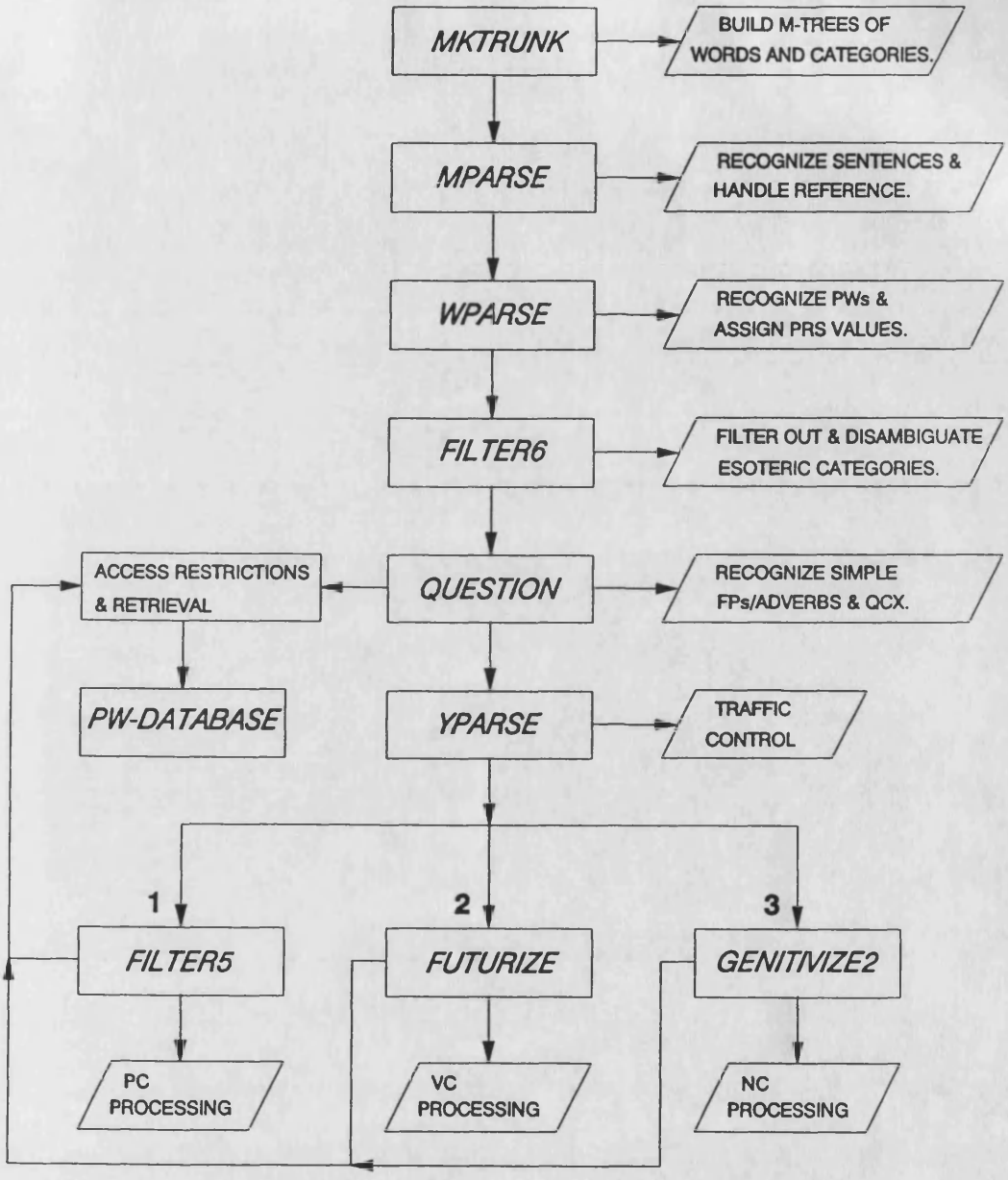


FIGURE 49: Top-level System Architecture in TUNIS1

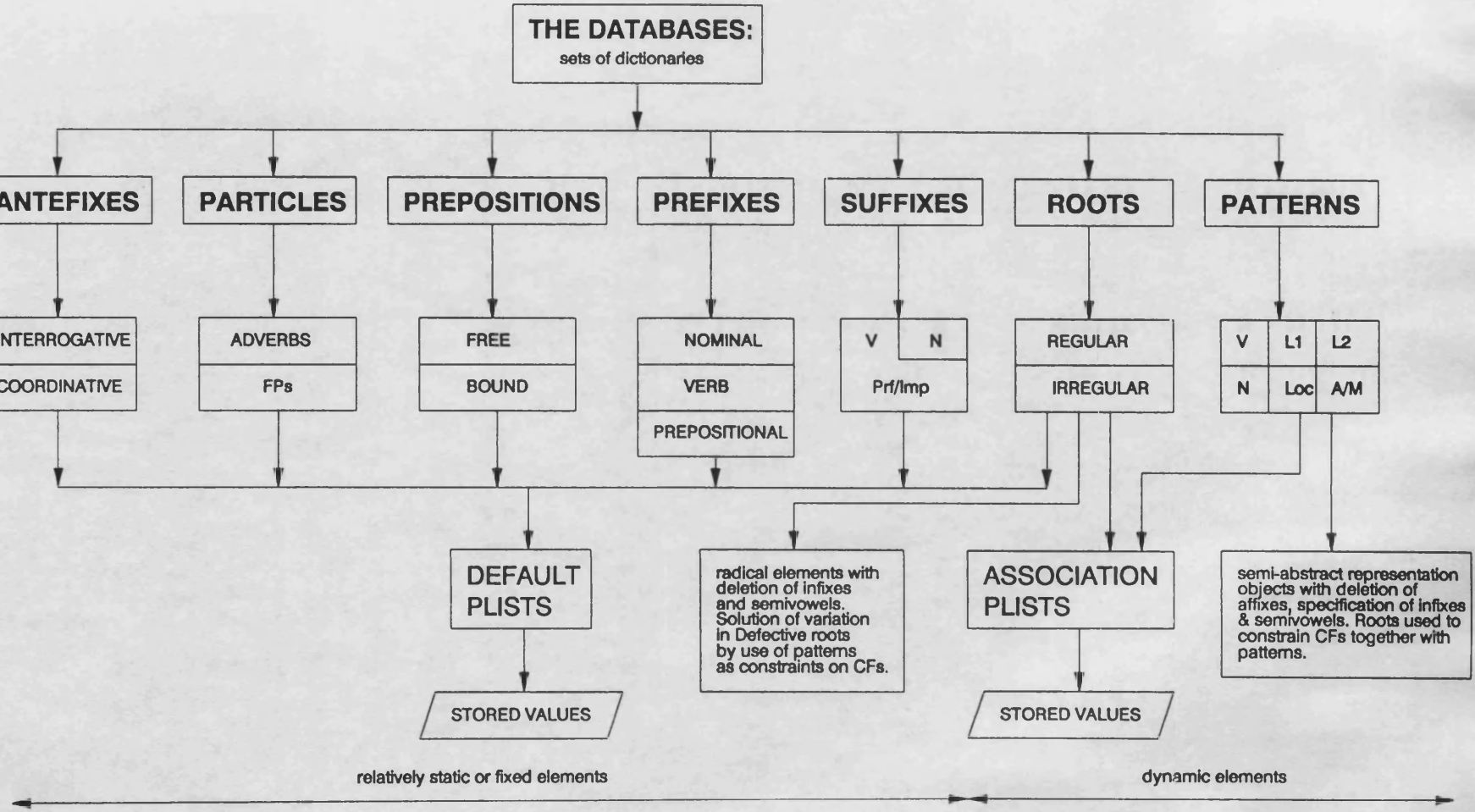


FIGURE 50: TUNIS1: The Logical Structure of the Databases

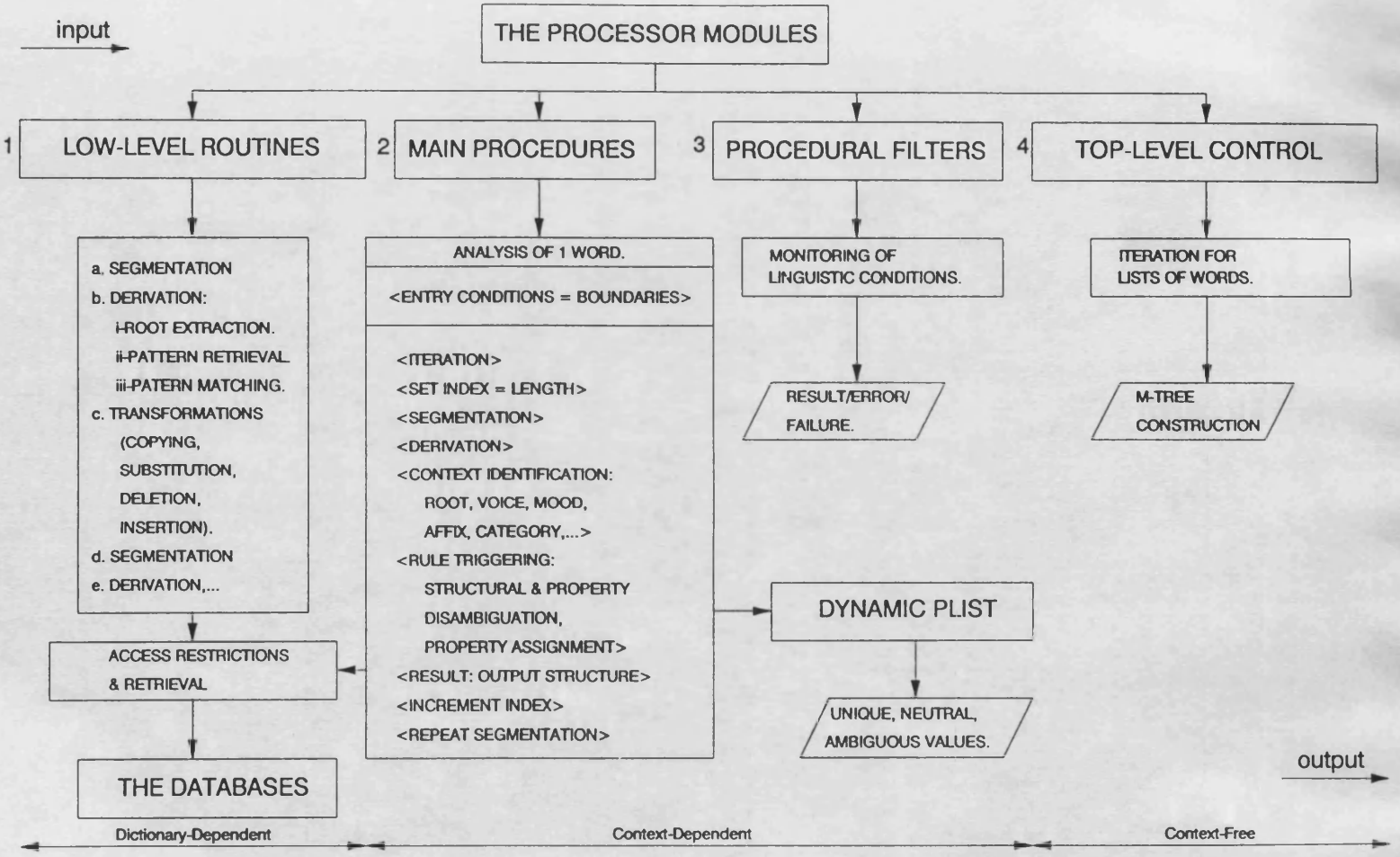


FIGURE 51: TUNIS1: The Logical Structure of the System

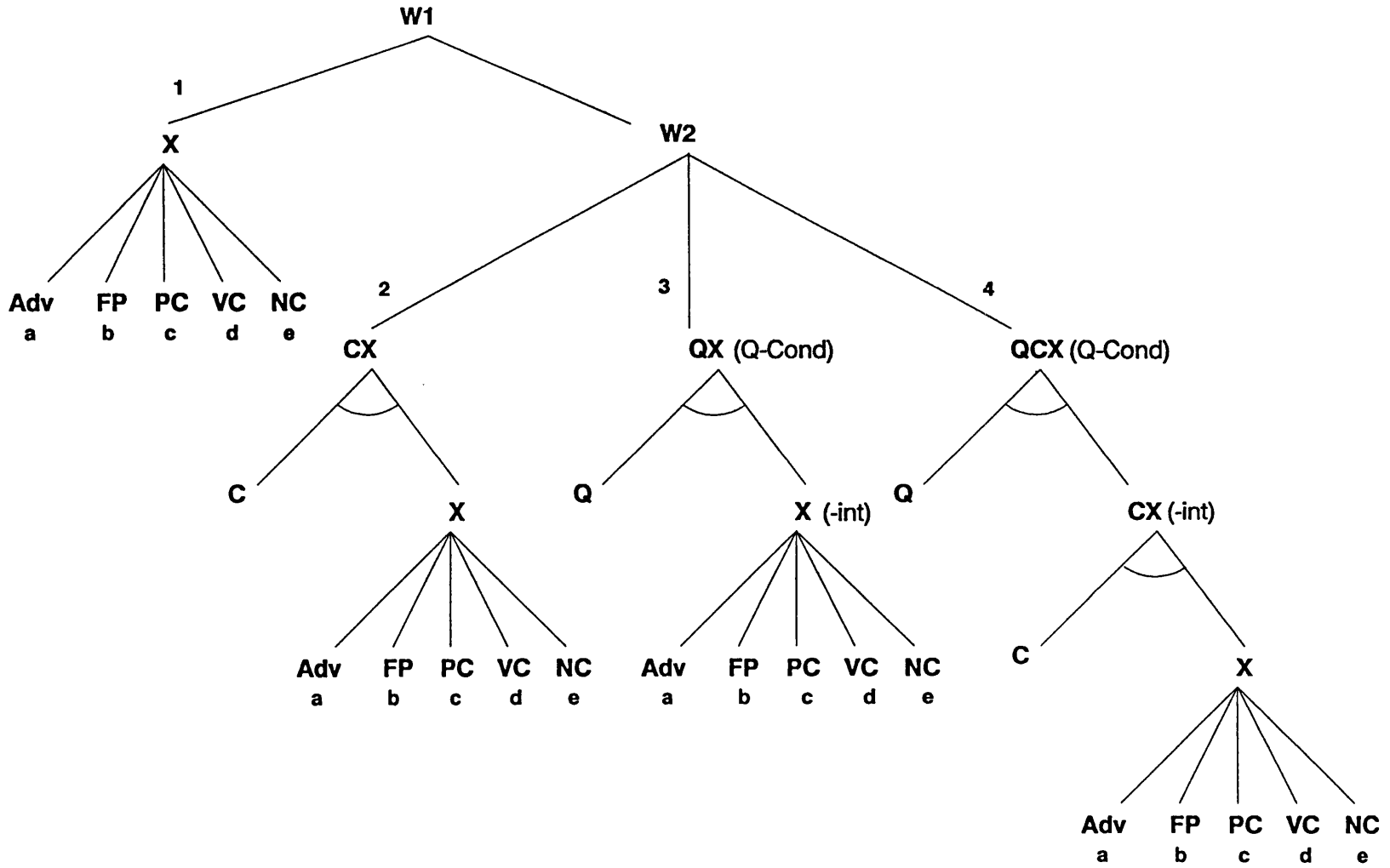


FIGURE 52: Search Paths and Constraints in a G-Tree for a Polysynthetic Word in M.S.A.

it then attempts to parse the word by invoking YPARSE and, if this FAILS, it reverts to left-to-right parsing by looking for Interrogative and Coordinative antefixes and, if it finds any, resumes dictionary look-up and processing in this order and in the same way as above.

The success of any dictionary find or parse following the identification of a Bound Interrogative Particle is determined by the satisfaction of the Q-Condition. This condition, namely that any morpheme X following Q must not be interrogative, is implemented as a constraint on the validation of a node that has already been solved. In the actual implementation, the PW-Processor also requires the satisfaction of constant-LENGTH conditions for the antefixes Q and C, of PW boundary conditions, and of the rule that any bound allomorphic variant for an FP cannot be in free variation, rather it has to be in complementary distribution with other affixes.

Further, idiosyncratic categorial constraints are implemented in the form of FILTER6 which either filters out or disambiguates output structures incoming from QUESTION. Figure 53 below illustrates the progress and direction of processing in the PW-Processor.

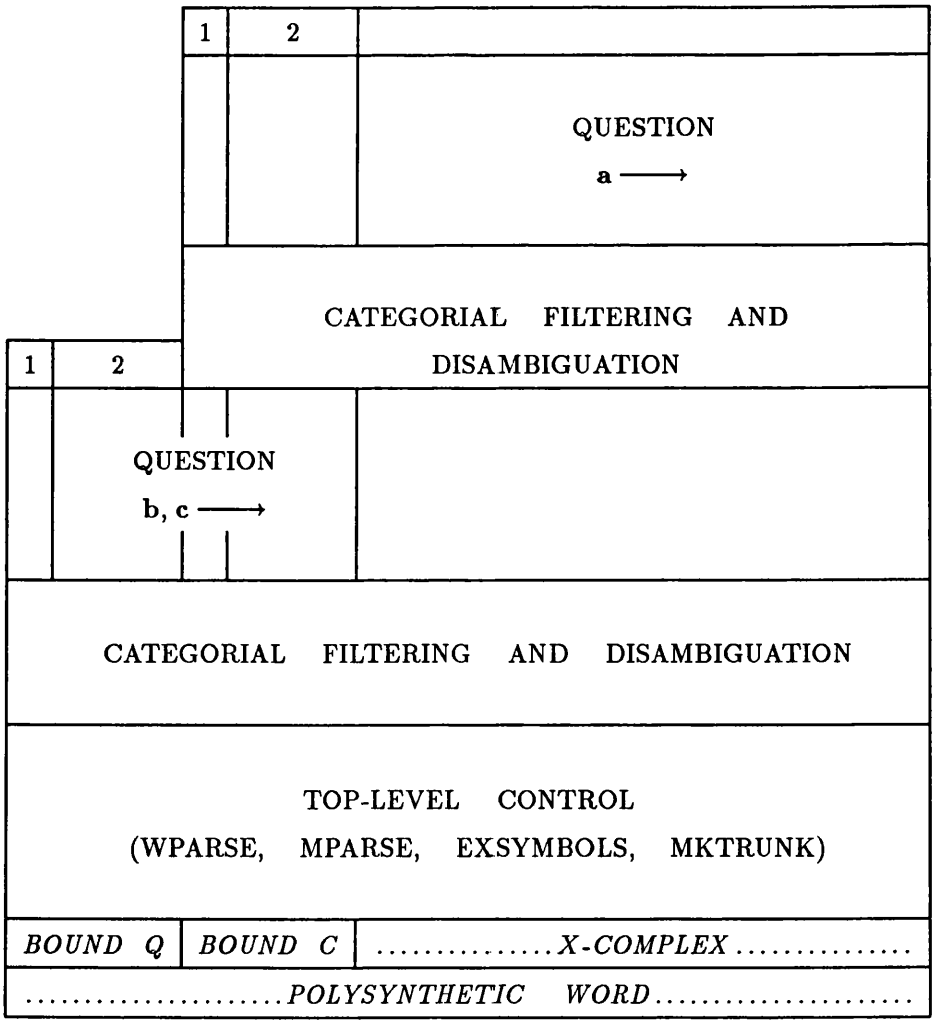


FIGURE 53: A Chart for the Direction of Parsing in the PW-Processor

Note that, here also, the PW-Processor generates flat M-Trees. From Figure 48 above, we can generate possible bracketed structures of the form: [(Q) (C) X]. However, only single or unified categorial descriptors—such as CX, for Coordinate PWs, and QX or QCX, for Interrogative PWs—are actually generated.

For further clarification of the procedures of the PW-Processor, we provide an automatic trace for the parse of a maximum-LENGTH Verb PW-Complex and some sample morphological parses by TUNIS1 where the function MKTRUNK invokes MPARSE as a subprogram (cf. Vol. II, App. D).

==0==<*****>==0==

Chapter 12

A LINGUISTIC AND COMPUTATIONAL EVALUATION OF TUNIS1

INTRODUCTION

In this Chapter, we aim to evaluate the linguistic validity and the computational performance of the TUNIS1 system. In particular, we ask the question: to what extent does the system anticipate syntactic and functional applications? In order to reach an answer to this question, we have to demonstrate the efficiency of the feature system described by the MA in solving potential syntactic problems. Specifically, the appraisal of this system involves the investigation of TRANSITIVITY, REFERENCE, GOVERNMENT, agreement, disagreement, ambiguity, and functional phenomena concerning Verb-complementation and Subject requirements, as well as the structural interface between morphological and syntactic categorization, and the generation of valid sentence trees in M.S.A. Such a demonstration should also provide us with a test-bed or platform for testing and evaluating the syntactic efficiency of the TUNIS1 system by suggesting prospective schemas or models and schedules for the formulation and implementation of syntactic parsers of Arabic.

In addition, we have to seek some means of assessing the computational performance of TUNIS1. Such an assessment has to include a critical account of the programming techniques used in this system and answer questions about its psychological realism, practicality, viability, and flexibility.

I TESTING THE LINGUISTIC VALIDITY OF THE SYSTEM: TEST SCHEMAS FOR THE FEATURE SYSTEM

The question we ask here is, to what extent does the feature system anticipate syntactic and functional analyses in M.S.A.? In order to answer this question, the first task we have to do is to suggest ways of expressing GOVERNMENT and agreement facts of Arabic as a means of putting to the test the feature values generated by TUNIS1. In the process, we introduce new formal notation and terminology for the discussion of these issues.

I.1 ASPECTS OF VERB AND NOMINAL GOVERNMENT IN M.S.A.

GOVERNMENT in Arabic syntax is a moot issue and not all Arab linguists are agreed on the significance of vowel endings in Verbs and Nominals. The ALBASRA school and their followers have always maintained that such vowels are either MOOD or CASE MARKS. For instance, ALSAYYID, 1982: Vol. I, 59–60, describes Arabic final vowels and suffixes in terms of MOOD and CASE values. However, the ALKUWFA school, as well as campaigners for the simplification of Arabic grammar, have simply dismissed the idea that such vowels could have any significance or function. For instance, ANIYS, 1951: 237, has it that "'case marks' never determined meanings in the minds of Arabs of old, as grammarians claim, rather they are no more than [*anaptyctic*] vowels needed mostly to create a [phonetic] liaison between words". Such positions stem from claims of the ALKUWFA school that there is no such phenomenon as GOVERNMENT in Arabic syntax and that there are no lexical or abstract governors. For example, ALMAXZUWMIY, 1966: 229, states this position clearly: "In reality, the argument for the adherence to the idea of government and its ensuing consequences such as resorting to logic and rationalizations to explain a structure or a construction, represent an era when the nature of linguistic study was not understood". ABDO, 1973: 21 and 26, also expresses this same position.

Whatever the merits of this argument propounded by the ALKUWFA school and their supporters, there are in fact observed correlations between certain free morphemes and the occurrence of particular inflectional suffixes in words that follow them. In order to satisfy the minimum of a descriptive-adequacy criterion, we have to account for these correlations, which we have indeed referred to as GOVERNMENT and described in Chapters 4, 6, and 8. However, we do not intend this account as an in-depth discussion but rather as a guide to using the feature system.

I.1.1 A Test Schema for TENSE and MOOD GOVERNMENT

In Chapter 4, § II.4.4, we dealt with questions pertaining to Particle GOVERNMENT of Verbs in TENSE and MOOD within CCFs containing affixed Future Particles. In Chapter 5, § I.2.3.7, these GOVERNMENT requirements for FCFs are implemented directly as conditions on the CFs in question, and we suggest that they can also be implemented in much the same way for non-affixed, or Free Particles followed by Verbs. For example, the Free Future Particle 'sawofa', "will, shall"—exactly like the Bound Future Particle 'sa', "going to"—governs Verbs in the imperfect TENSE and indicative MOOD. Other Particles, as listed in Chapter 8, may have different TENSE and MOOD requirements.

In general, Arabic Free Particles governing Verbs can be subdivided into three classes, according to their MOOD GOVERNMENT requirements, as follows:

a. *if* X = [+indicator] $\Longrightarrow$ MDG (X) = [+ind];

b. *if* X = [+subjunctor] $\Longrightarrow$ MDG (X) = [+sub]; and

c. *if* X = [+jussor] $\Longrightarrow$ MDG (X) = [+jus].

Using the above definitions, and given a Free Particle P and a Verb V, we can summarize Particle-Verb GOVERNMENT with the following generalizations:

P πm V *IFF* P governs V in TENSE and MOOD.

We can now express a prototype or prospective schedule for Particle-Verb GOVERNMENT in M.S.A.:

d. *if* P πm V $\Longrightarrow$

i. *if* P = 'qado', "perhaps, already" $\Longrightarrow$ *if* V [+imp] $\Longrightarrow$ V [+ind], else V [+prf];

ii. *if* P [+indicator] $\Longrightarrow$ V [+imp] and V [+ind];

iii. *if* P [+subjunctor] $\Longrightarrow$ V [+imp] and V [+sub];

iv. *if* P [+jussor] $\Longrightarrow$ V [+imp] and V [+jus];

v. else MDG (P) = MMK (V) and V [+ind].

I.1.2 A Test Schema for DEFINITENESS, FLEXION, and CASE GOVERNMENT

I.1.2.1 Particle-Nominal GOVERNMENT

In Chapter 6, § II.1.7, we discussed the process of genitivization whereby Bound Prepositions are affixed to and govern CNFs in the oblique (i.e, genitive/prepositional) CASE. In Chapter 7, § I.2.3.12, we implemented this CASE requirement directly as an affixation condition. Here again we propose that this condition, as it applies to Free Particles, can be expressed in a parallel way to that used for Bound Particles. For instance, the Free Preposition 'fiy', "in"—similarly to the Bound Preposition 'bi', "at, in ..."—governs Nominals in the oblique CASE. Other Particles, listed in Chapter 8, have accusative CASE GOVERNMENT, and might also have requirements for specific DEFINITENESS values in the Nominal following them. Hence, in CASE GOVERNMENT terms, there are two kinds of Arabic Free Particles governing Nominals:

a. *if* X = [+governor] = GVT (X) = [+ACC]; and

b. *if* X = [+inflector] = GVT (X) = [+OBL].

Given P as a Free Particle and N as a Nominal, we can then express Particle-Nominal GOVERNMENT in Arabic, as follows:

P πc N *IFF* P governs V in CASE and DEFINITENESS.

Using these definitions, we can formulate a prototype schedule for Particle-Nominal GOVERNMENT in M.S.A. as follows:

c. *if* P πc N $\Longrightarrow$

i. *if* P = 'la|', "no, not" $\Longrightarrow$ N [+ndf] and N [+ACC];

ii. *if* P [+governor] $\Longrightarrow$ N [+ACC];

iii. *if* P [+inflector] $\Longrightarrow$ N [+OBL];

iv. else GVT (P) = CMK (N).

I.1.2.2 Inter-Nominal GOVERNMENT

In addition to Particle-Nominal GOVERNMENT in Arabic, there is also GOVERNMENT between two adjacent Nominals in that the first of them modifies the second in CASE value. This type of GOVERNMENT falls under two CATEGORIES:

a. A GENITIVIZATION process called ':id;a|fat', "annexation" which has two general subcategories:

i. An indefinite Nominal or a Nominal with ambiguous DEFINITENESS followed by another indefinite Nominal as in: 'kita|bu rajuli+', "a man's book".

ii. An indefinite Nominal or a Nominal with ambiguous DEFINITENESS followed by a definite one as in: 'kita|bu :alrrajuli', "the man's book".

In C.A., there is another type of annexation in which a definite Nominal is followed by another definite Nominal. This type is rare in M.S.A.

b. A SPECIFICATION, or 'tamoyiyz', process whereby a Nominal is specified as to nature, size, quantity, type, structure, etc. by a second Nominal as in 'xamosuwna kita|ba+', "fifty books".

Given two Nominals $N1$ and $N2$, the statements in *a.* and *b.*, can be generalized as follows:

a.i. $N1 \quad \pi a \quad N2 \quad \mathcal{IFF} \quad N1$ [+ndf]/$N1$ [+xdf] $\Longrightarrow$ $N2$ [+ndf] and $N2$ [+OBL].

a.ii. $N1 \quad \pi b \quad N2 \quad \mathcal{IFF} \quad N1$ [+ndf]/$N1$ [+xdf] $\Longrightarrow$ $N2$ [+dfp] and $N2$ [+OBL].

b. $N1 \quad \pi s \quad N2 \quad \mathcal{IFF} \quad N1$ [+nnf] $\Longrightarrow$ $N2$ [+fvm] and $N2$ [+ndf] and $N2$ [+ACC].

I.2 ASPECTS OF GOVERNMENT AND (DIS)-AGREEMENT IN M.S.A.

Agreement in Arabic syntax is another complex issue which is often characterized by controversy and acrimonious debate. In M.S.A., agreement between two elements of sentence structure is subject to word order, i.e., which of the two elements precedes the other. Here again, not all Arab linguists are agreed on the necessity or requirement for agreement conditions to apply between sentence constituents. For instance, ANIYS, 1951: 214, argues that in Verbal sentences

> "the imperfect [Verb] hardly ever undergoes any changes as it does not agree with the subject in duality or plurality. It may agree with it in feminine gender, with true-feminine Nouns, but this agreement is not of such necessity and obligation as that which takes place when the Subject precedes [...]. Thus, the agreement of the Verb with its Subject is not as stringent as that of the Adjective with the Noun, which is required to follow it in feminine gender, duality, and plurality".

However, ABDO, 1973: 70, is in disagreement with ANIYS. He advances, loc. cit., arguments to support the thesis known as: (the Arabic dialect of ':akaluwniy :alobara|giyt-', "they-bit-me the-insects"), a derogatory reference to certain schools, mainly the ALKUWFA, which claim that there is complete agreement between Verbs and Subjects with the Verb preceding. AL-SAYYID, 1982: Vol. I, 271, gives examples of such a dialect where Verbs agree in NUMBER with their Subjects, despite the precedence of the former; and he gives, ibid. 274–275, other examples where preceding Verbs and following Subjects disagree in GENDER. However, he comments, ibid. 271–275, that such examples represent the exception rather than the rule, which, in M.S.A., requires that there should be NUMBER disagreement (that is all Verbs

should be singular whether the Subject is singular, dual or plural) and GENDER agreement between a Verb and a Subject following it.

Unfortunately, we cannot do justice, in this Chapter, to such involved issues. However, our aim is to suggest a representative model for handling agreement issues and highlight some syntactic problems that are in need of solutions. The precise details of this model can be refined in an eventual adequate theory of Arabic Syntax, something which is beyond the scope of this discussion.

I.2.1 A Test Schema for NGP and DFC (Dis)Agreement Correlated with HUMANITY Values

I.2.1.1 Subject-Verb NGP (Dis)Agreement

We have not seen examples of agreement in the course of this thesis, but we had occasion (in Ch. 5, I.1.2.*c*.) to give examples relating to the resolution of homonymic MOOD values, and to Subject-Verb agreement. In M.S.A., there are two standard types of Subject-Verb agreement:

a. GENDER-only agreement for Verbs which precede Subjects, with the Verb frozen in third-PERSON-singular form irrespective of the NUMBER of the following Subject, as in 'jalasa :alo:awola|du', "sat 3 (M) S the boys", "the boys sat". This example supports GREENBERG's 1966: 94, Universal 33 (where N and V stand respectively for Noun and Verb): "when number agreement between the N and V is suspended and the rule is based on order, the case is always one in which the V precedes and the V is in the singular".

b. NUMBER, GENDER, and PERSON agreement for Verbs which follow the Subject provided that, if this Subject is plural, it should be human as well. Non-human plural Subjects disagree with following Verbs in that the latter are frozen in feminine-third-PERSON-singular form. An example of the former is ':alo:awola|du jalasuw|, "the boys sat 3 (M) P", and an example of the latter is: ':alo:aqola|mu takassarato', "the pens broke 3 (F) S".

We give below a table with new notation for expressing generalized NGP agreement configurations, including homonymic NGP values; and given the definitions in Table 61, and given N as a Nominal Subject and V as a Verb, we can generalize *a*. and *b*. above in the following schedule for Subject-Verb NGP (dis)agreement correlated with HUMANITY values:

c. $V\ \Omega\ N\quad \mathcal{IFF}$

i. *if* N [+pl] $\Longrightarrow$

- *if* N [+hm] $\Longrightarrow$ V [+sn] and V [+r3] and $V\ \Omega G\ N$, and

RULE TYPE	NGP VALUES REQUIRED
X ØΩN Y	*if* X [+dp] $\Longrightarrow$ Y [+dl]/Y [+pl]; or *if* Y [+dp] $\Longrightarrow$ X [+dl]/Y [+pl]; or *if* X [+sp] $\Longrightarrow$ Y [+sn]/Y [+pl]; or *if* Y [+sp] $\Longrightarrow$ X [+sn]/X [+pl].
X ΩN Y	*if* NBR (X) = NBR (Y); or *if* X ØΩN Y.
X ØΩG Y	*if* X [+mf] $\Longrightarrow$ Y [+mc]/Y [+ff]; or *if* Y [+mf] $\Longrightarrow$ X [+mc]/X [+ff].
X ΩG Y	*if* GDR (X) = GDR (Y); or *if* X ØΩG Y.
X #G Y	*if* GDR (X) $\neq$ GDR (Y); or *if* X ØΩG Y.
X ØΩP Y	*if* X [+rx]/[+xr] $\Longrightarrow$ Y [+r2]/Y [+r3]; or *if* Y [+rx]/[+xr] $\Longrightarrow$ X [+r2]/X [+r3].
X ΩP Y	*if* PRS (X) = PRS (Y); or *if* X ØΩP Y.
X ΩGP Y	*if* X ΩG Y and X ΩP Y.
X ΩNG Y	*if* X ΩN Y and X ΩG Y.
X ΩNGP Y	*if* X ΩNG Y and X ΩP Y.

TABLE 61: Generalized NGP (Dis)Agreement Configurations

- *if* N [−hm] $\Longrightarrow$ V [+sn] and V [+r3] and V [+ff]; or

ii. *if* N [−pl] $\Longrightarrow$ V ΩNGP N.

d. $N\ \Omega\ V$ $\mathcal{IFF}$

i. *if* N [+pl] $\Longrightarrow$

- *if* N [+hm] $\Longrightarrow$ N ΩNGP V, and
- *if* N [−hm] $\Longrightarrow$ V [+sn] and V [+r3] and V [+ff]; or

ii. *if* N [−pl] $\Longrightarrow$ V ΩNGP N.

I.2.1.2 Nominal-Adjective NG and DFC (Dis)Agreement

In M.S.A., Adjectives and Nominals (dis)agree along similar lines to Subjects and Verbs, the common factor being the requirement of HUMANITY values as a determinant of (dis)-agreement. However, besides NUMBER and GENDER (dis)agreement, Adjectives always agree with Nominals in DEFINITENESS, FLEXION, and CASE. For example, in ':awola|du+kiba|ru+', "big boys", there is agreement between the Nominal and the following Adjective in that both are indefinite, and have full vowel marks, and nominative CASE MARKS. The

Nominal and the Adjective in ':alomudarrisuwna :alo:awa|:ilu', "the first teachers" are both definite and nominative, but the first has a full NGC suffix while the second has a single vowel mark. This is simply another type of FLEXION agreement between Nominals. This FLEXION agreement is as important as any of the other features although not picked up by any of the Arab Grammarians. For instance, ALMAXZUWMIY, 1966: 187, states that Arabic Adjectives agree with Nominals in "definiteness, gender, number, and case" with no mention of FLEXION.

Furthermore, the two examples given above also show NGP agreement since the two Nominals there are human. However, in: 'kutubu+ t-amiynatu+', "expensive books", where 'kutubu+' is non-human, there is NG disagreement between the Nominal and the Adjective.

Before we proceed to the statement of a prototype schedule for Nominal-Adjective (dis)-agreement, here is a table with new notation for expressing generalized DFC agreement configurations, including homonymic values for DFC properties.

RULE TYPE	DFC VALUES REQUIRED
X ΩD Y	*if* DFN (X) = DFN (Y); or *if* X [+xdf] $\Longrightarrow$ Y [+dfp].
X ΩF Y	*if* X [+fvm]/X [+nnf] $\Longrightarrow$ Y [+fvm]/Y [+nnf]; or *if* X [+svm] $\Longrightarrow$ Y [+nnf]; or *if* X [+xvm] $\Longrightarrow$ Y [+fvm]/Y [+nnf]; else FLXN (X) = FLXN (Y).
X ΩC Y	*if* X [+nmm] $\Longrightarrow$ Y [+ncp]/Y [+npm]; or *if* X [+acm] $\Longrightarrow$ Y [+ncp]/Y [+cpm]; or *if* X [+obm] $\Longrightarrow$ Y [+ncp]/Y [+cpm]/Y [+npm]; or *if* X [+ncp] $\Longrightarrow$ Y [+nmm]/Y [+acm]/Y [+obm]; or *if* Y [+ncp] $\Longrightarrow$ X [+nmm]/X [+acm]/X [+obm]; else CMK (X) = CMK (Y).
X ΩDC Y	*if* X ΩD Y and X ΩC Y.
X ΩDFC Y	*if* X ΩDC Y and X ΩF Y.

TABLE 62: Generalized DFC (Dis)Agreement Configurations

Given the definitions in Tables 61 and 62, and given N as a Nominal and A as an Adjective, we can generalize from the examples above a schedule for Nominal-Adjective DFC (dis)agreement correlated with HUMANITY values as follows:

N Ω A $\mathcal{IFF}$ N ΩDFC A, and

a. *if* N [+pl] $\Longrightarrow$

i. *if* N [+hm] $\Longrightarrow$ N ΩNG A; or

ii. *if* N [−hm] $\Longrightarrow$ $N \not\equiv G\ A$ and A [+sn]; and

b. *if* N [+dp]/N [+sp] $\Longrightarrow$ $N\ \emptyset\Omega N\ A$ and $N\ \Omega G\ A$; and

c. *if* N [+sn]/N [+dl] $\Longrightarrow$ $N\ \Omega NG\ A$.

Note that the above schedule takes into account homonymic NUMBER values, as in the example of 'nah-ona', "we", which agrees with dual or plural Adjectives, as well as taking account of homonymic FLEXION values, as in the example of 'mabonay', "building", which agrees with Adjectives with full vowel marks as in 'mabonay kabiyru+', "a big building", but can also be a single-voweled Modifier as in 'mabonay :alo:amoja|di', "the building of glories".

I.2.2 A Test Schema for NG and DFC (Dis)Agreement and GOVERNMENT Correlated with NUMERICAL VALUES

Nominal-Numeral agreement is indeed another thorny area where there is little agreement about agreement, disagreement or GOVERNMENT between Nominals. HASSAN, 1978: Vol. IV, 517, sums up this state of affairs: "there is a myriad of conflicting rules on offer in this [Nominal-Numeral (dis)agreement] topic as well as much disagreement and contradiction about it".

While most Arab linguists are agreed on NUMBER agreement between Nominals and Numerals, the polemic has continued to centre around GENDER (dis)agreement between them. Early Arab Grammarians, especially from the ALBASRA school, espouse the GENDER-disagreement position. For instance, SIYBAWAYHI, 1970: Vol. II, 176, opines that "for Nouns that are masculine in the singular and that are quantified with a Numeral that denotes a number greater than two and up to ten, the Numeral itself has to be feminized with a feminine suffix whereas for Nouns that are feminine in the singular [and modified with the same Numeral], the Numeral is feminine although devoid of feminine suffixes". ALBUSTAANIY, 1977: 71, 83, 255, 321, 393, and 395, argues that Numeral forms without feminine suffixes are feminine nonetheless, and that Numeral forms with feminine suffixes are masculine nonetheless. Thus, he repeatedly refers, loc. cit., to such Numeral forms as "deviating from the norms" and being "extraneous to regularization".

However, the general trend is to support the thesis for GENDER disagreement between Nominals and Numerals if the Numeral precedes. For instance, HASSAN, 1978: Vol. IV, 537, states that "for disagreement to take place, there are two provisos: that the Noun be present in the text and that it follow the Numeral". If the Numeral is postposed then YAEQUWB, 1986: 355, exhorts that "both gender agreement and disagreement are permitted [...] but conforming to the [disagreement] rule is preferred". A similar position is reiterated in HASSAN, 1978: Vol. IV, 537-8.

Our aim here is to give a foretaste of the question of Nominal-Numeral (dis)agreement, but for a thorough discussion of this topic the reader is referred to HASSAN, 1978: Vol. IV, 517–567. We shall now concentrate on the use of NUMERICAL VALUES in formulating a prototype schedule for that (dis)agreement. First, here are a few instances showing two broad categories of such (dis)agreement.

a. For Numerals preceding Nominals, there are three possibilities:

i. If the Numeral has a NUMERICAL VALUE between three and ten, then it disagrees with the Nominal in NUMBER and GENDER, and governs it in the oblique CASE, as in: 't-ala|t-atu kutubi+', "three books".

ii. If the Numeral has a NUMERICAL VALUE in the set: {20, 30, 40, 50, 60, 70, 80, 90}, then it agrees with the Nominal in NUMBER, it has neutral GENDER agreement with the Nominal, and governs it in the accusative CASE, as in: 'ɛisvoruwna binota+', "twenty girls".

iii. If the Numeral has a NUMERICAL VALUE in the set: {100, 200, 1000, 2000}, then the Nominal is in the singular and has neutral GENDER agreement with the Numeral which governs it in the oblique CASE, as in: ':alofayo rajuli+', "2000 men".

b. For Numerals following Nominals, there are four possibilities:

i. If the Nominal is singular or dual then the Numeral has to agree with it in NUMBER and GENDER, as in: ':alokita|bu :alowa|h-idu', "the one book".

ii. If the Nominal is plural and the Numeral has a NUMERICAL VALUE between three and ten, then the Numeral has a singular form and disagrees in GENDER with the Nominal, as in: ':alrrija|lu :aloxamosatu', "the five men".

iii. If the Nominal is plural or homonymic between plural and dual or between plural and singular and the Numeral's VALUE is in the set: {20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 1000, 2000}, as in: 'nah-onu :aloɛisvoruwna', "us twenty", or if the Nominal is homonymic between dual and plural and the Numeral's VALUE is bounded between two and ten, as in: 'nah-onu :alo:it-ona|ni/:alo:it-onata|ni', "us two", then there is neutral GENDER agreement between the Nominal and the Numeral and the Numeral's NUMBER is unspecified.

iv. If the Nominal is homonymic between singular and plural, then if the Numeral's VALUE is one, then the NUMBER of the Numeral is singular and there is GENDER agreement between the two as in: ':alomabonay :alowa|h-idu', "the one building", and 'mudarrisiy :alowa|h-idu', "my one teacher", and if the Numeral's VALUE is bounded between three and ten, then the NUMBER of the Numeral is singular and there is GENDER disagreement

between the two, as in: 'mudarrisiy (zayodi+) :aloxamosatu', "the five teachers of (Zayd)".

In addition and similarly to Nominal-Adjective agreement, in *i.–iv.* above, Nominals agree with Numerals in DFC.

Given the notation introduced so far and the definitions in Tables 61 and 62, and given N as a Nominal and M as a Numeral, we can now express *a.* and *b.* above as a generalized schedule for Nominal-Numeral NG and DFC (dis)agreement and GOVERNMENT correlated with NUMERICAL VALUES as follows:

c. Given a sequence $M \quad N$, and m as the NUMERICAL VALUE of M, then,

i. *if* $m = [3, 10] \implies$

M [+sn] and N [+pl] and M #G N, and $M \quad \pi$a $\quad N$ or $M \quad \pi$b $\quad N$;

ii. *if* $m = \{20/30/40/50/60/70/80/90\} \implies M$ [+pl] and N [+sn] and

M ØΩG N and $M \quad \pi$s $\quad N$; and

iii. *if* $m = \{100/200/1000/2000\} \implies M$ [+sn]/M [+dl], and N [+sn] and

M ØΩG N and $M \quad \pi$a $\quad N$ or $M \quad \pi$b $\quad N$.

d. Given a sequence $N \quad M$, and m as the NUMERICAL VALUE of M, then,

N ΩDFC M, and

i. *if* N [+sn]/N [+dl], and $m = [1/2] \implies N$ ΩNG M;

ii. *if* N [+pl] and $m = [3, 10] \implies M$ [+sn] and N #G M;

iii. *if* N [+pl]/N [+dp]/N [+sp], and

$m = \{20/30/40/50/60/70/80/90/100/200/1000/2000\}$; or

if N [+dp] and $m = [2, 10] \implies N$ ØΩ M; and

iv. *if* N [+sp], then

if $m = [1] \implies M$ [+sn] and N ΩG M; and

if $m = [3, 10] \implies M$ [+sn] and N #G M.

I.3 ASPECTS OF HOMONYMY AND DISAMBIGUATION

I.3.1 Homonymic Feature Values in TUNIS1

We have seen that TUNIS1 yields a rich feature system with three kinds of values: unique, neutral, and ambiguous. We argued in Chapter 5, I.1.2.*c.*, that neutral values offer natural flexibility for a syntactic parser in that lexical items with such values can be made to agree with other items that may have indifferent or unspecified values for the same feature. In this Section (Ch. 5, I.1.2.*c.*), as well as in the discussion so far, we have shown how neutral and ambiguous values are to be incorporated in schedules for (dis)agreement and GOVERNMENT in M.S.A. In general, we can say that:

a. neutral values agree or disagree with unspecified values for a given feature.

b. GOVERNMENT applies "vacuously" to lexical items with neutral values: such items are taken to satisfy GOVERNMENT conditions per se.

c. with regard to ambiguous values, the strategy we can adopt is to predict positions in sentences where they can crop up and proceed to disambiguate them, then subsequently apply agreement and GOVERNMENT conditions.

In the Section below, we give a few examples of disambiguation in sentence contexts which again draw up parallels with the disambiguation techniques followed in morphological-concatenation contexts.

I.3.2 A Test Schema for Disambiguation in Syntactic Contexts

In the course of the program description, we saw some examples of homonymy in NUMBER, HUMANITY, REFERENCE, CATEGORY, and DFC, as well as in other values. Since homonymy in one feature is often correlated with homonymy in other features, resolving this homonymy for one feature can often lead to disambiguating the others. For instance, resolving homonymic categorial values may lead to disambiguation in other features. A prototype schema based on categorial homonymy resolution in syntactic contexts is given below.

Given a lexical item X with categorial homonymy, a Nominal N, an Adjective A, a Numeral M, and a Verb V, then

a. *If* CAT (X) = WN, e.g., 'mudarrisiy', "my teacher/(the) teachers (of) ....", then

i. *if* X / [ ] X N [+OBL] $\Longrightarrow$

CAT (X) = MD, and NBR, HTY, and DFC (X) = pl, hm, ndf, dnf, cpm; and

ii. *if* X / [ ] X A, or *if* X / [ ] X N [+dfp] $\Longrightarrow$

CAT (X) = AN, and NBR, REF, and DFC (X) = sn, col, dfp, svm, ncp.

b. *If* CAT (X) = VM, e.g., 'ɛasvara', "ten/join a group of ten", then

i. *if* X / [ ] X N [+NOM], or *if* X / [ ] PD X $\Longrightarrow$ CAT (X) = VB; and

ii. *if* X / [ ] M X $\Longrightarrow$ CAT (X) = MM.

c. *If* CAT (X) = VN, e.g., 'walada', "give birth/(the) son (of)", then

i. *if* X / [ ] X N [+NOM], or *if* X / [ ] PD X $\Longrightarrow$ CAT (X) = VB; and

ii. *if* X / [ ] X N [+OBL], or *if* X / [ ] AP X $\Longrightarrow$ CAT (X) = MD.

d. *If* CAT (X) = WV, e.g., 'h-asabahu', "he counted it/his honour", then

i. *if* X / [ ] X N, or *if* X / [ ] PD X $\Longrightarrow$ CAT (X) = VB; and

ii. *if* X / [ ] X V, or *if* X / [ ] AP X $\Longrightarrow$ CAT (X) = AN.

e. *If* CAT (X) = VA, e.g., ':axod;ara', "green/become green", then

i. *if* X / [ ] SP X $\Longrightarrow$ CAT (X) = VB; and

ii. *if* X / [ ] VB X $\Longrightarrow$ CAT (X) = MD.

f. *If* CAT (X) = XA, e.g., ':uwlay', "first", then

i. *If* X / [ ] N X $\Longrightarrow$ CAT (X) = AA, and FLXN (X) = fvm; and

ii. *if* X / [ ] X N [+OBL] $\Longrightarrow$ CAT (X) = MD, and FLXN (X) = svm.

g. *If* CAT (X) = XN, e.g., 'mabonay', "a building/(the) building (of)", then

i. *if* X / [ ] X A $\Longrightarrow$ CAT (X) = NN, and FLXN (X) = fvm; and

ii. *if* X / [ ] X N [+OBL] $\Longrightarrow$ CAT (X) = MD, and FLXN (X) = svm.

We have stated that the disambiguation of homonymic values in *a.* through to *g.* above depended on categorial-homonymy resolution, but we can also see, in the above examples, that this resolution often stems from syntactic facts of Arabic such as those described so far. For instance, having stipulated a rule: P πm V, and specified the GOVERNMENT requirements of a particular Particle, we can suggest conditions such as in *b.i.*, *d.i.*, and *e.i.* above, where a homonymic VM, WV, or VA preceded by a Verb Particle PD or SP must be interpreted as a Verb and not as MM, AN, or MD. Thus, 'qado h-asabahu', can only be "already, he counted it", and not "!already, his honour". Similarly, in *c.ii.* and *d.ii.*, a homonymic VN or WV preceded by a Nominal Particle AP must be interpreted as a Nominal and not as a Verb, as a result of the requirement that P πc N. Thus, ':inna h-asabahu', can only be "indeed, his honour",

and not "!indeed, he counted it". Again, the rules $N1$ πa $N2$, and $N1$ πb $N2$, with $N2$ being required to carry oblique CASE, allow us in *a.i.*, *c.ii.*, *f.ii.*, and *g.ii.*, to disambiguate a homonymic WN, VN, XA, or XN as MD and not as AN, VB, AA, or NN. Hence, 'mudarrisiy zayodi+', can only be "the teachers of Zayd", and not "!my teacher, Zayd". The situation, in *a.ii.*, results from the agreement rule $N\ \Omega\ A$. Hence, 'mudarrisiy :alo:awwalu', can only be "my first teacher", and not "!the teachers of the first". These heuristics, as well as others which we will refer to below, can be useful in resolving several types of homonymic examples referred to in Chapters 4–11.

I.4 ASPECTS OF WORD ORDER IN M.S.A.

Besides the descriptions given for GOVERNMENT, (dis)agreement and disambiguation in M.S.A. syntax, we need to define a structural level where such phenomena can be tested. In order to move towards such a level, the first necessary task is to provide a mechanism for mapping morphological CATEGORIES into syntactic terms or CATEGORIES.

I.4.1 From Morphological to Syntactic Categorization: The M-S Interface

We have seen that given a Verb CATEGORY, TUNIS1 generates morphological-CATEGORY descriptors, of the type: VB, VR, PVB, PVR, QVB, QVR, CVB, CVR, QPVB, QPVR, CPVB, CPVR, QCPVB, and QCPVR. If we take into account the GOVERNMENT rules in § I.1.1.1 above, we realize that such descriptors can indeed be useful since they are different in syntactic distribution.

For example, given the rule: P πm V, we can rule out a syntactic structure: *P PVB. Similarly, given the rule: P πc N, we can rule out the sequence: *P PNN. Thus, '*sawofa sa:akotubu', "*I will am-going-to-write", and '*fiy bi:alomakotabati', "*in at the library", are ungrammatical structures. Taking into account the Q-Condition referred to in Chapter 10, II.1.*i.*, we can rule out sequences such as *QPD QVB and *QPr QDN as in: '*:aqado :akataba?', "*Q-already did-he-write?" and '*:aɛinoda :azayodu+?', "*Q-with is Zayd?".

In order to control these distributional aspects, we need an interface which maps morphological CATEGORIES into syntactic CATEGORIES. Such an interface can simply be a rewriting rule schema with syntactic symbols on the left-hand side and morphological symbols on the right-hand side, as follows:

F ⟶ FD/QFD/CFD/QCFD ...

B ⟶ SP/QSP/QCSP ...

$P \longrightarrow$ Pr/QPr/QCPr ...

$A \longrightarrow$ AA/DA ...

$M \longrightarrow$ MM/DM ...

$L \longrightarrow$ L1/L2/QL1 ...

$V \longrightarrow$ VB/PVB/VR/QVR ...

$N \longrightarrow$ NN/DN/AN/QNN ...

We have specified in Tables 57–59 the possible morphological CATEGORIES for each class of word. It is suggested that such tables can be used to expand the above M-S interface using the same method. Having defined an M-S interface, we can then specify distributional conditions on the non-terminal left-hand side of each M-S interface rule. Again, such conditions can take the form of a rewriting-rule system. For instance, we can have a rule for rewriting Verb structures in Arabic such that a Verb "sequence" is simple *V*, *B* followed by VB, *B* followed by VR, *F* followed by VB, or *F* followed by VR, etc., thereby excluding *SP PVB, *FD PVR, etc.

I.4.2 The Syntactic Expression Level: Adjacency Rules

We have seen that the description of affixation conditions implies an Annotated Grammar with rewriting-rules for Arabic morphology. Similarly, the syntactic account provided so far with its descriptions of GOVERNMENT, (dis)agreement conditions, and disambiguation contexts, in effect implies an Annotated Grammar for the rewriting of Arabic syntactic structures or sequences. Here, we have carefully avoided the use of the term *phrase*, since we do not wish to subscribe to any particular theory of grammar at this juncture. Instead, we shall simply use the word *expression* to refer to a single unit of structure inside a sentence, the important factors here being the precedence or order of related words inside such a unit and the conditions relating them to each other. The other facet that is not implied here—and that ensues from the term *phrase*—is that of hierarchy.

Early Arab transformationalists such as ALKHUWLIY, 1981: 113, FAAXUWRIY, 1980: 18–20, and ZAKARIYYA, 1982: 143, (as well as SNOW, 1965: 29, who calls his VP a *PRED*, KILLEAN, 1966: 42, 45, et passim, and LEWKOWICZ, 1967: 214), have insisted on the existence and desirability of hierarchical concepts such as *VPs*, (i.e., Verb Phrases), in Arabic. Other generativists, such as AOUN, 1979b, have even gone so far as to suggest a "discontinuous VP" in Arabic. However, CHOMSKY himself, 1981: 127–128, concedes that "a crucial assumption has been that there is a category VP in the X-bar system of the base, thus permitting GFs [i.e., Grammatical Functions] to be defined in terms of structural configurations. But there are languages in which this does not seem to be true, Classical Arabic, with VSO structure, is a

case in point.". This position is clearly stated in SAAD, 1982: Vol. II, 11, "therefore Arabic has no VP". It is also a position adopted by ALFEHRI, 1985: Vol. I, 105, fn. 4, "I have tried, in my research, to prove [...] that Arabic does not have a VP constituent".

Assuming this proviso for non-hierarchical structure in Arabic, we can now give some examples for expression rules that we can also call *adjacency rules*. Given a Verb eXpression, VX, a Nominal eXpression NX, a Prepositional eXpression PX, and ( )* as a notation for optional recursive structures, we can envisage expression rules of this type:

a.i. VX ⟶ V;

a.ii. VX ⟶ F VB/VR and F πm V; or

a.iii. VX ⟶ B VB/VR and F πc V.

b.i. NX ⟶ N $(A)*$ and N ΩNG A and N ΩDFC A;

b.ii. NX ⟶ N M and N Ω M and N ΩDFC M;

b.iii. NX ⟶ M NN/DN and M Ω N and M π N; or

b.iv. NX ⟶ MD (MD)* NN/DN and MD π N.

c.i. PX ⟶ PGP/PNN/PDN/PAN ...; or

c.ii. PX ⟶ P NX and P πc NX.

Such rewriting rules can serve as a model for the description and the implementation of syntactic rules in M.S.A. However, there is a degree of redundancy in rules like *a.i.*, since they will generate structures of the type: [VX [V [VB . kataba]]]. This problem can be solved at the level of implementation simply by flattening trees of this kind to eliminate the pursuit of trivial paths. The precise definition of a flattening procedure is a computational issue which, due to constraints of scope, we do not consider here. The implementation of the agreement and GOVERNMENT conditions, in *a.–c.* above, can follow a parallel method to that suggested for the application of affixation conditions to M-Trees.

I.4.3 The Generation of Sentence-Rewrite Rules: From M-Trees to S-Trees

We argued above for a set of adjacency rules with expression elements being assigned either to precedence or to secondary positions. These precedence rules happen to coincide with GREENBERG's position 1966: 73–87 and 108, which classifies Arabic as a "type I/Pr/NG/NA language", type I being a VSO order, Pr a Prepositional type, NG a Nominative-Genitive order, and NA a Nominative-Adjective order.

However, the issue of word order in Arabic is far from settled. In recent years, and following typological and generative studies, the polemic surrounding Arabic word order has been refuelled. In particular, the moot questions are: to what extent this order is free, and whether Arabic is a VSO or an SVO language. The question being of relevance to generativists, such as

ALFEHRI, since it determines which order is derived from which base, or underlying, structure.

Early Western transformationalists such as SNOW, 1965: 6, et passim, KILLEAN, 1966: 41, and LEWKOWICZ, 1967: 136–138, 164–167, 219, et passim, have argued for an SVO order in Arabic, following earlier linguistic trends to claim the universality of the rule [S [NP VP]], although LEWKOWICZ, 1971: 815,fn. 10, explains that she assumes as basic the SVO order (which she now notes as [S [NP Pred]]) because "I am familiar [sic] with it". However, the early trend in Arabic linguistics was to support the thesis for VSO. Hence, ANIYS, 1951: 243, argues for VSO order and SAAD, 1982: 8 and 11, embraces the VSO position. In addition, BAKIR, 1980: xi, defends the VSO hypothesis and ALFEHRI, 1982: 40 and 90, and 1985: Vol. I, 105, argues for a VSO order based on generative argumentation in terms of movement rules and so on. TAHHAAN, 1972: Vol. II, 54–55, and ABDO, 1973: 71, had taken the SVO position as early as the seventies and against the trend, referred to above, of Arabic linguistic research at that time.

It is, however, interesting that both ANIYS supporting the VSO theory and ABDO supporting the SVO one, have taken two positions that are poles apart following the same line of argument: a dismissal of GOVERNMENT and any CASE values in Arabic inflection suffixes, a position expressed in ABDO, 1973: 97–111, and ANIYS, 1951: 243, and which leads to the attempt at imposing a base order in order to be able to determine Subjects and Objects, this being impossible in Arabic without CASE MARKS and word order. Hence, on the question of the extent of freedom of word order ANIYS, 1951: 295, pronounces "every language obeys a given system of word order and this order is observed in sentence construction". This position is espoused by ABDO, 1973: 71, to support the opposite SVO thesis where he gives the following example without specified CASE MARKS, ':alrrajul d;[a]r[a]b :alowalad', "the man hit the boy", and claims, loc. cit., that: if this example "was said in any [sic] language in the world, the Subject will not be any other than ':alrrajul', 'the man'".

Nevertheless, DEGACHI, 1984: 4 and 177, and 1989: 87–88, questions the relevance of typological data in formulating an adequate theory of word order, and presents some arguments for a relatively free-word-order theory of Arabic, restrictions being imposed, inside expressions by precedence rules and outside expressions by agreement and GOVERNMENT conditions, as described above.

However, our aim here is to suggest ways of expressing Arabic syntactic rewriting rules which can be implemented in an eventual syntactic parser. ALJURJAANIY (d.1078), 1981: 64, illustrates clearly the difficulties arising from freedom of word order in Arabic by giving the following possible examples: 'zayodu+ munot;aliqu+, zayodu+ yanot;aliqu, yanot;aliqu zayodu+, munot;aliqu+ zayodu+, zayodu+ :alomunot;aliqu, :alomunot;aliqu zayodu+, zayodu+ huwa :alomunot;aliqu, and zayodu+ huwa munot;aliqu+', where Zayd is a Proper Noun, 'yanot;aliqu', "departs", is an imperfect Verb, 'munot;aliqu+', "departing" and ':alomunot;aliqu', "the de-

parting", are Verbals, and 'huwa', "he, himself", is a Subject Pronoun.

These examples show extreme flexibility in the order of Arabic sentences including Verb and Nominal sentences and, although dating back a few centuries, could actually be sentences of M.S.A. Such examples we suggest, can be accounted for by a CSG with annotations, such as the grammar below, where $\sim$ stands for free order:

$S \longrightarrow$ NX $\sim$ VX and NX Ω VX;

$S \longrightarrow$ NX $\sim$ L and NX ΩNG L; or

$S \longrightarrow$ NX $\sim$ A and NX ΩNG A.

I.4.4 TRANSITIVITY, Functional Aspects, and REFERENCE

The grammar given in Chapter 12, § I.4.3, is not sufficient to generate "all and only" valid sentences of Arabic. We need to extend this grammar so that we can take account of functional phenomena in Arabic syntax. For instance, for every Arabic sentence which has a Verb and a Subject, the Subject must be in the nominative CASE. ALSAYYID, 1982: Vol. I, 14–15, gives two curious examples which are at variance with this generalization, and where the Subjects are in the accusative and the Objects in the nominative: 'xaraqa :alt-t-awobu :alomisoma|ra', "the nail tore the dress", and 'kasara :alzzuja|ju :aloh-ajara', "the stone broke the glass". Nevertheless, ALSAYYID, loc. cit., comments that these examples are exceptional. Further, it is often possible to omit the Subject in Arabic sentences, if it can be understood from the context, as in: 'ja|:a', "(he) arrived", but if the Verb is transitive and in third PERSON, then the Subject is obligatory. Hence, '*d;araba :alowalada', "*hit the boy", is ungrammatical, whereas 'd;arabotu :alowalada', "(I) hit the boy", is valid. These rules, can be expressed as a Subject requirement condition such as in *a.* below:

a. *If* $S \longrightarrow \ldots V \ldots$; then

i. *if* V [−ntr] and V [+r3] $\Longrightarrow$ $S \longrightarrow \ldots V \sim N$ [+NOM] ... ; where N is an obligatory nominative Subject.

ii. else $S \longrightarrow \ldots V \sim (N\ [+NOM])$ ... ; where N is an optional nominative Subject.

In addition, we have to introduce some specification for Object and Complement requirements implied by the TRANSITIVITY values described in our program. Some examples are given in *b.* below:

b.i. *If* V [+mtr] $\Longrightarrow$ $S \longrightarrow \ldots N$ [+ACC]; and N = Object.

b.ii. *If* V [+dtr] $\Longrightarrow$ $S \longrightarrow \ldots N1$ [+ACC] ... $N2$ [+ACC]; and

$N1$ and $N2$ = Objects.

b.iii. *If* V [+ptr2] $\Longrightarrow$ $S \longrightarrow \ldots N$ [+ACC] ... PX ... ; and

N and PX = Objects.

b.iv. *If* V [+ctr] $\Longrightarrow$ $S \longrightarrow \ldots$ AA [+ACC] ... ; or

$S \longrightarrow \ldots$ PX ... ; and AA or PX = Complements.

Furthermore, a rule is needed to solve the REFERENCE in sentences like ':alowaladu d;arabahu zayodu+', "the boy hit-him Zayd" where the referential Pronoun 'hu', "him" is identical with "the boy". This IDENTITY is expressed in NUMBER, GENDER, and PERSON agreement and could be formulated as a REFERENCE rule such as in *c.* below:

c. There is identity (X p) *IFF* X ΩNGP p, where p is a referential Pronoun and X is an antecedent.

i. *If* $S \longrightarrow \ldots Y\$p \ldots$; where p is a Pronoun attached to Y, then

- *if* p [+exl] $\Longrightarrow$ $S \longrightarrow \ldots X \ldots Y\$p \ldots$; where X is an obligatory antecedent of p and identity (X p) holds; and
- *if* p [+col] $\Longrightarrow$ $S \longrightarrow \ldots (X) \ldots Y\$p \ldots$; where X is an optional antecedent of p and identity (X p) holds *if* X.

Such rules as Subject, Object, and Complement requirements, as well as REFERENCE resolution will be essential to the generation of valid Arabic sentences. They can also be useful in disambiguating homonymic values for a given lexical item. For example, the conditions in I.3.2.*b.i.* and *c.i.*, above, stem from the Subject requirement rule, whereas the conditions in I.3.2.*a.ii.*, *b.ii.*, *e.i* < *i.*, *f.*, and *g.* above, ensue from the facts of precedence described in this Chapter (§ I.3.2 and § I.4.2). However, the vital question to be answered here is in which order the various syntactic rules are to be applied. In our view, the best strategy in computational terms is to exclude invalid sequences as early as possible in parsing, the advantage being to avoid unnecessary work if the sentence is going to be eventually rejected as ungrammatical. It is hoped that the above discussion demonstrates how the feature system can be useful in doing just that.

II TESTING THE COMPUTATIONAL PERFORMANCE OF THE SYSTEM

II.1 THE LINGUISTIC CONTRIBUTION TO THE EFFICIENCY OF THE SYSTEM

Throughout this presentation, we have seen what we might term a practical approach to computational analysis rather than advocating a purely theoretical approach based on the elegance of one formalism or another. This practicality or realism of the TUNIS1 system is represented in its linguistic approach to computation, an approach which is based on the principle of generating "all and only" valid morpholcgical sequences. In order to achieve this end, the program uses an array of techniques such as below:

a. Entry conditions whose aim it is to "nip in the bud" sequences that do not observe boundary conditions, thus reducing parsing time.

b. Effective use of LENGTH constraints as conditions on the progress of a given parse.

c. Efficient use of linguistic constraints expressing MOOD and CASE GOVERNMENT, TRANSITIVITY requirements, and Graphotactic Conditions of affixation which filter out ungrammatical sequences and favour the rejection of invalid results as early as possible.

d. Efficient use and organization of the dictionaries means that dictionary-dependent rules can be used to constrain rule application, such as the PROMOTion of TRANSITIVITY values based on intensive- and causative-root distinction in the database.

e. The adoption of pattern matching in preference to generative transformations means that vowel and semivowel specification together with root insertion can be used as constraints on pattern realizations and saves the expensive use of a generative transformational component, besides solving the complexity of Arabic morphology and especially with respect to Defective roots.

f. The adoption of a "non-autonomous" morphological analysis means that this analysis can and does have access to a graphological level represented by rules such as compatibility, assimilation, and elision, as well as a semantic level represented by rules such as a NUMERICAL VALUE ASSIGNment, and a syntactic level with ASSIGNments for TRANSITIVITY requirements. We have seen that such levels of analysis can be useful for instance in early resolution of homonymy and thus, rather than argue for the elegance of "Autonomous Morphology", we have striven for a practical solution to the problems at hand.

All these linguistic features have led us to a conditioned approach to computational analysis. This conditioned approach gives the system a deterministic flavour, in that it is predictive. However, there is some flexibility in the parsing procedures represented in the use of filters which control overgenerated sequences and impose secondary conditions on idiosyncratic exceptions. We have seen that such exceptions are often too intricate and would interrupt the flow of parsing

if incorporated in the main procedures.

II.2 STRUCTURAL ASPECTS OF THE SYSTEM

The TUNIS1 system makes use of the concept of search trees as plans for action represented in Goal Trees with obligatory or choice routes, and offering a natural expression for the formulation of the above conditions. The search method adopted is a modified depth-first cut-off approach, in preference to a blind context-free search. The optimization of the search does not only come from conditioning the paths, but also from the adoption of heuristics which favour the right-to-left approach with fewer affixes to parse on the right, and the strategy of investigating the minimal number of paths first, which implies that the search starts with simple words, such as Adverbs and Free Particles, and progresses towards more complex concatenations, such as (P-, V-, and N-) Complexes.

Besides search strategies and heuristics, TUNIS1 is characterized by an independent modular structure which is preferred to the rigidity of linear programming. This makes for a clarity of structure which is useful in debugging and updating functions.

==0==<*****>==0==

CONCLUSION

I PRACTICAL SYNTHESIS

Throughout the computational analysis carried out in this thesis, we found that Cambridge LISP is particularly well-suited for the processing of Arabic morphology and serves to enhance the efficiency of our morphological parser. The efficiency of TUNIS1 is also enhanced by its extensive testing, including syntactic assessment, and by the simplicity of pattern-matching techniques which allow TUNIS1 to deal with Defective Verbs and Nominals and general variation in Arabic words in an efficient way and without resorting to the use of complex generative transformations. For example, our lexicon includes 522 entries but the parser can recognize an estimated 90,000 different word forms. Further, by returning error messages after rejecting invalid words, TUNIS1 offers the user the advantage of informative clues which can be useful in teaching.

In addition, the linguistic conditioning and the search strategies (described in Ch. 12) favour a reduction in parsing time and, therefore, saving in computational costs. Previously (Ch. 2, § III.2.1), we defined a primary objective: demonstrating the validity of our linguistic approach; and a secondary objective: achieving a viable timing on morphological parses, which we defined as within one minute per long complex sentence. The results actually achieved in this respect are indeed positive. Table 63 below gives sample timing results from morphological parses of different kinds of words, of grammatical sentences, and of randomly chosen groups of words (as listed in App. D and indicated with the phrase "used in Table 63"). The results include an equal number of slowest and fastest times for the function MKTRUNK.

II THEORETICAL SYNTHESIS

The discussion in Chapter 12, § I.1–I.4, has attempted to give a foretaste of topical issues in Arabic syntax, but the main objective in introducing these issues was to demonstrate the flexibility and usefulness of the feature system used, not only for morphological analysis, but

TYPE	TIME in SECONDS	AVERAGE per word
1 word	2.56	
1 word	3.52	
1 word	0.06 (fastest)	
1 word	0.18	
1 word	0.26	
1 word	0.22	
1 word	0.4	
1 word	5.4	
1 word	7.32	
1 word	8.44 (slowest)	2.83
1 sentence (2 words)	2.1 (slowest)	1.05
1 sentence (2 words)	0.2 (fastest)	0.1 (fastest)
1 sentence (3 words)	3.58 (slowest)	1.19
1 sentence (3 words)	0.44 (fastest)	0.14
1 sentence (4 words)	3.38 (slowest)	0.84
1 sentence (4 words)	0.42 (fastest)	0.1
1 sentence (5 words)	4.22 (slowest)	0.84
1 sentence (5 words)	0.86 (fastest)	0.17
1 sentence (6 words)	5.52 (slowest)	0.92
1 sentence (6 words)	2.12 (fastest)	0.35
1 sentence (7 words)	9.44 (slowest)	1.34
1 sentence (7 words)	3.6 (fastest)	0.51
1 sentence (8 words)	11.1 (slowest)	1.38 (slowest)
1 sentence (8 words)	9.69 (fastest)	1.21
1 random group (30 words)	39.08	1.30 (slowest)
1 random group (28 words)	28.28	1.01
1 random group (26 words)	10.02	0.38 (fastest)
MEAN AVERAGE		
per word		1.48
per sentence		4.04

TABLE 63: Sample Timing Results for M-Parsing in TUNIS1

also in anticipating syntactic processing. Such anticipation can be seen in the usefulness of NUMERICAL-VALUE correlations with (dis)agreement and GOVERNMENT phenomena, the

use of morphological categorization and ambiguity flags in syntactic disambiguation, the use of property labels such as those for TRANSITIVITY in functional sentence requirements and those for HUMANITY in (dis)agreement rule application, as well as the use of M-Trees as input to the interface between Morphology and Syntax. A side effect of this discussion was to introduce new notation and terminology for the analysis of syntactic facts of M.S.A.

However, there are undoubtedly further features which could be added in order to extend the coverage of our eventual syntactic processor. For instance, we could refine the selectional criteria for Object requirements by introducing a feature [+countable] to be ASSOCIATed with Objects required by Verbs like 'h-sb', with the meaning "to count". Other features that could be useful are the distinction of Collective and non-Collective Nominals. There are also extensions that may be desirable for our grammar, such as *Head Selection* criteria in given expressions in Arabic, as well as a host of other grammatical phenomena that we have not covered here, such as (dis)agreement between compound Numerals and Nominals, and between ordinal Numerals and Nominals. Further, we need to cover in our eventual grammar, other types of sentence such as imperative, relative, coordinative, conditional, vocative, and interrogative sentences.

The agreement rules can be made more flexible, if we wish to do so, owing to the flexibility of the property labels. For instance, we might wish to account for examples like 'ja|:ato :alrrija|lu', "the men came" where 'ja|:ato' disagrees in NUMBER and GENDER with ':alrrija|lu'. A rule to describe this example could be applied exceptionally, i.e., where such disagreement is allowed, such that V #NG N, instead of applying the general rule (described in Ch. 12, I.2.1.1.*c.i.*). In addition, we have to find methods for allowing dynamic variation in the use of categorial morphological descriptors in syntax. For instance, a Verbal L1 or L2 could be used as a Nominal or as an Adjective. However, the grammar fragment described in Chapter 12 was not intended to be exhaustive. Rather, it was an eclectic analysis meant to illustrate the feature system used, to perform extensive testing for the analysis, to show a parallelism between the way we have analysed morphology and the way we could analyse syntax, and to serve as a prototype for an adequate theory of arabic syntax.

The implementation of such a theory has to decide on a formalism (e.g., ATN, DCG, etc.) that might best represent the theory. Further, the implementation has to find ways of defining ordering functions to handle strict- and free-order sequences, optional and obligatory functional requirements, recursive structures, (dis)agreement, GOVERNMENT, REFERENCE, and disambiguation rules and conditions. For instance, a decision has to be made on whether an eventual syntactic processor of Arabic should proceed in a *top-down* fashion with backtracking, for ambiguous cases—techniques which are suited for *fixed-word-order* languages, and which characterize BEN HAMADOU's 1986b system; or whether a *bottom-up* approach might be more appropriate for *free-word-order* languages. An alternative and useful technique was suggested

by DEY, 1986: 65–67, in which conditioned loops are traversed until all functional roles, treated as a set, have been filled. Such an approach, proposed for Hindu—a relatively free-word-order language—seems to coincide with the linguistic approach advocated here, although, DEY, 1986, is based on a conventional ATN formalism.

Besides the development of a syntactic component for the processing of Arabic by computer, prospects for further research also include the expansion of the database, the development of an interface which would translate our transcription's output (after doing the parsing) into an Arabic font, to develop components for the computational analysis of semantic and pragmatic aspects of Arabic, and eventually to develop practical applications, such as Machine Translation, error detection and spelling, and Computer-Assisted Teaching.

The precise implications of our approach for a purely theoretical analysis of Arabic morphology, as well as situating our morphological approach within a wider theory of Arabic language structure, remain to be investigated in further research. However, we hope that above all we have demonstrated the importance of computational linguistics as a testing ground for linguistic theories, the need for computerization in Arabic linguistics, as well as demonstrating the following points:

1) that the listing approach (proposed mainly in WP models) is inefficient in accounting for the morphological data of Arabic.

2) that traditional Arabic approaches to MA are inexplicit and insufficient for the analysis of Arabic morphology and that traditional analyses (such as those of IBNU JINNIY, c.1913, and SIYBAWAYHI, 1970) fragment morphology between phonology and syntax.

3) that generative approaches to morphology (such as those proposed by BAKALLA, 1979, and ANDERSON, 1982) are inadequate in also fragmenting MA between phonology, syntax, and the lexicon and remain rather dismissive of Semitic morphology.

4) that the results achieved by TUNIS1 are positive in terms of yielding a MA which is both computationally efficient and theoretically sound, uniform, and coherent and that we have laid the basis for future work in this field.

5) that our approach is superior to combinatorial methods of MA (such as those proposed by ALBAWWAAB et al., 1989; DEBILI and ZOUARI, 1989; BEN HAMADOU, 1986a and 1986b), and could fruitfully form the basis for further work in this field.

==0==<*****>==0==

BIBLIOGRAPHY

A BIBLIOGRAPHIC COMMENTS AND ABBREVIATIONS

The bibliography is divided into two parts: 1) Arabic titles listed in English alphabetical order (with titles transcribed exactly, using my transcription, and the authors, their publishers, and places of publication transcribed loosely, using a simplified version of my transcription, in order to facilitate reading; the names of Arab authors who have published in non-Arabic languages being written as published); and 2) non-Arabic titles listed in English alphabetical order. The Computational Linguistics journal was formerly: The American Journal of Computational Linguistics. The bibliography also uses the following abbreviations:

[1]	*Actes du IVème Colloque International de Linguistique: Linguistique Arabe et Informatique*, Tunis, 9–12 Novembre, 1987. CERES, Université de Tunis, Tunisie.
[2]	*Proceedings of the First Conference on Bilingual Computing in Arabic and English*, Cambridge, 6–7 September, 1989.
[3]	*Proceedings of the Second Conference on Bilingual Computing in Arabic and English*, Cambridge, 5–7 September, 1990.
[4]	*Proceedings of the Eleventh International Conference on Computational Linguistics (COLING 86)*, Bonn, 25–29 August, 1986. ICCL.
ACL	Association for Computational Linguistics.
ACM	The Association for Computing Machinery.
ALECSO	Arab League Educational Cultural and Science Organization.
ALPAC	Automatic Language Processing Advisory Committee.
CERES	Centre d'Etudes et de Recherches Economiques et Sociales.
CSLI	Center for the Study of Language and Information.
CUP	Cambridge University Press.
Ed.	Editor.
ICCL	International Committee on Computational Linguistics.
M.A.	Master of Art.
MIT	The Massachusetts Institute of Technology.
Ph.D.	Doctor of Philosophy.
Procs.	Proceedings.
n.r.p.	no running page numbers.
sup.	supplement.

B ARABIC WORKS

ALECSO (1989) ':alomajohuwda|t :allatiy bad-alatoha| :alomunad/d/amat fiy mayoda|n :alo-:iɛola|miyyat litakuwn ɛawon ɛalay xidomat :allugat :aloɛarabiyyat watat;owiyr :alomujo-tamaɛ :aloɛarabiyy', (sup. to) [1], 1987: 53 pp.

Abdo, D. (1973) *:aboh-a|t- fiy :allugat :aloɛarabiyyat.* Printed by Dar Alqalam Press Co, Beirut, Lebanon.

Abu Almakaarim, A. (n.d.) *taqowiym :alofikor :alnnah-owiyy.* Dar Alt-t-aqa|fat, Beirut, Lebanon.

Albakkuwsh, A. (1973) *:alttas;oriyf :aloɛarabiyy min xila|l ɛilom :alo:as;owa|t :aloh-adiyt-.* Alsvsvarikat Alttuwnisiyyat Lifunuwn Alrrasm, Tunis, Tunisia.

Albakkuwsh, A.; Alsouissi, R.; and Ben Hamadou, A. (1989) 'muɛojam :alomusot;alah-a|t :allisa|niyyat :alo:iɛola|miyyat', in [1], 1987: 139–166 (Arabic Section).

Albawwaab, M.; Miyr Alam, Y.; and Altayyaan, M. H. (1989) 'nid/a|m :isvotiqa|q :alokalimat :aloɛarabiyyat bi:aloh-a|sib', in [1], 1987: 25–63 (Arabic Section).

Albustaaniy, B. (1977) *muh-iyt; :alomuh-iyt;, qa|muws mut;awwal lillugat :aloɛarabiyyat.* Maktabat Lubna|n, Beirut, Lebanon.

Aldahdaah, A. (1981) *muɛojam qawa|ɛid :allugat :aloɛarabiyyat fiy jada|wil walawoh-a|t.* Maktabat Lubna|n, Beirut, Lebanon.

Alfehri, A. A. (1985) *:allisa|niyya|t wa:allugat :aloɛarabiyyat, nama|d;ij tarokiybiyyat wadala|liyyat.* Volume I. Dar Tubqa|l Lilnnasvr, Casablanca, Morocco. 1st Edition.

Ali, N. (1988) *:allugat :aloɛarabiyyat wa:aloh-a|suwb.* Alkhat Press, Cairo, Egypt.

Aljamɛiyyat Alɛilmiyyat Almalakiyyat Alurduniyyat, Dairat Alhaasib Al:ilaktruwniyy. (1985) *dira|sat fanniyyat h-awol :alttarojamat :alo:a|liyyat fiy :alowat;an :aloɛarabiyy.* ALECSO, Tunis.

Aljurjaaniy, A. (1981) *dala|:il :alo:iɛoja|z.* Ridha, M. R., Ed. Nasvr Dar Almaɛrifat, Beirut, Lebanon.

Aljurjaaniy, Ali. I. M. A. (1978) *kita|b :alttaɛoriyfa|t.* Maktabat Lubna|n, Beirut, Lebanon.

Alkhuwliy, M. A. (1981) *qawa|ɛid tah-owiyliyyat lillugat :aloɛarabiyyat.* Dar Almirriyx Lilnnasvr, Riyadh, Saudi Arabia. 1st Edition.

Almaxzuwmiy, M. (1966) *fiy :alnnah-ow :aloɛarabiyy, qawa|ɛid watat;obiyq ɛalay :alomanohaj :aloɛilomiyy :alh-adiyt-.* Must;afay Alba|biy Alh-alabiy Wa:awola|duh, Cairo, Egypt. 1st Edition.

Almisriyy, A. (1983) ':isvoka|liyyat :allisa|niyya|t :aloh-adiyt-at, waqa|:iɛ :alnnadowat :allatiy ɛuqidat fiy :alddaworat :alssa|disat lillisa|niyya|t', in *:alofikor :aloɛarabiyy :alomuɛa|s;ir.* Volume XXV, March–April, 1983: 154–159. Markaz Al:inma|: Alqawmiyy, Beirut, Lebanon.

Alraajih-iy, A. (1983) *:alttat;obiyq :alnnah-owiyy.* Dar Alnnahdat Alɛarabiyyat Lit;t;iba|ɛat Walnnasvr, Beirut, Lebanon.

Alsaaqiy, F. M. (1977) *:aqosa|m :alokala|m :aloɛarabiyy, min h-ayot- :alsvsvakol wa:alowad/iyfat.* Nasvr Maktabat Alxa|njiy, Cairo, Egypt.

Alsamarraaiy, I. (1982) *:aloɛarabiyyat tuwa|jih :aloɛas;or.* Dar Alh-urriyyat Lit;t;iba|ɛat, Baghdad, Iraq.

Alsayyid, A. A. (1982) *fiy ɛilom :alnnah-ow.* Volume I. Dar Almaɛa|rif, Cairo, Egypt. 5th Edition.

Alɛaqabiy, M. (1984) ':alttarojamat :alo:a|liyyat, :ad;owa|: ja|nibiyyat', in *:alobaya|n.* Volume XXIV, May–June, 1984: 90–93. Kuwait.

Aniys, I. (1951) *min :asora|r :allugat.* Maktabat Alanglo Almisriyyat, Cairo, Egypt. 1st Published, 1951. Reprinted, 1958, 1971, 1978.

Bakiyr, A.; Almhiriy, A.; Nagra, T.; and Ben Aliyyat, A. (1974) *:alnnah-ow :aloɛarabiyy min xila|l :alnnus;uws; litala|mid-at :alssanat :alrra|biɛat min :alttaɛoliym :alt-t-a|nawiyy, nah-ow :alomaɛa|niy.* Publications du Bureau de l'Education, Tunis.

Bakiyr, A.; Almhiriy, A.; Nagra, T.; and Ben Aliyyat, A. (1975) *:als;s;arof :aloɛarabiyy litala|mid-at :alssanat :alo:uwlay wa:alt-t-a|niyyat wa:alt-t-a|lit-at min :alttaɛoliym :alt-t-a|na-*

wiyy. Société Tunisienne de Diffusion, Tunis.

Bakiyr, A.; Almhiriy, A.; Nagra, T.; and Ben Aliyyat, A. (1977) *:alnnah-ow :aloɛarabiyy min xila|l :alnnus;uws; litala|mid-at :alssanat :alo:uwlay min :alttaɛoliym :alt-t-a|nawiyy*. Société Tunisienne de Diffusion, Tunis. 4ème Edition.

Bakiyr, A.; Almhiriy, A.; Nagra, T.; and Ben Aliyyat, A. (1983) *:alnnah-ow :aloɛarabiyy min xila|l :alnnus;uws; litala|mid-at :alssanat :alt-t-a|niyyat min :alttaɛoliym :alt-t-a|nawiyy*. Société Tunisienne de Diffusion, Tunis. 4ème Edition.

Dak Albaab, J. (1982) 'madoxal :ilay :allisa|niyya|t :aloɛa|mmat wa:aloɛarabiyyat—:alomanohaj :alowas;ofiyy :alowad/iyfiyy', in *:alomawoqif :alo:adabiyy*. Volumes CXXXV-CXXXVI, 1982: 42–64. Mat;a|biɛ Alif Ba|: Al:adiyb, Damascus, Syria.

Dar Almashriq. (1968) *:alomunojid :alo:abojadiyy*. Almat;baɛat Alka|t-uwlikiyyat, Beirut, Lebanon. 2nd Edition.

Dar Almashriq. (1986) *munojid :alt;t;ulla|b*. Edited by Albustaaniy, F. I. Almat;baɛat Alka|-t-uwlikiyyat, Beirut, Lebanon. 31st Edition. 1st Published, 1941.

Degachi, A. (1989) 'taqoyiym :alnnad/ariyya|t :alomat;oruwh-at lilomuɛa|lajat :alo:a|liyyat lillugat :aloɛarabiyyat wa:isotiɛoma|l :alnnah-ow :alobunoyawiyy :alomuwah-h-ad kah-all li:isvoka|liyyat tarotiyb :aloɛana|s;ir', in [1], 1987: 65–111 (Arabic Section).

Dhayf, S. (1979) *:alomada|ris :alnnah-owiyyat*. Dar Almaɛa|rif, Cairo, Egypt. 4th Edition.

Faaxuwriy, A. (1980) *:allisa|niyyat :alttawoliydiyyat :alttah-owiyliyyat*. Mansvuwra|t Lubna|n Aljadiyd, Beirut, Lebanon. 1st Edition.

Hassaan, A. (1979) *:allugat :aloɛarabiyyat, maɛona|ha| wamabona|ha|*. Mat;a|biɛ Alhay:at Almisriyyat Alɛa|mmat Lilkita|b. Cairo, Egypt. 2nd Edition.

Hassan, A. (1978) *:alnnah-ow :alowa|fiy*. Volume IV. Dar Almaɛa|rif, Cairo, Egypt. 4th Edition.

Ibnu Jinniy, A. U. (1979) *:allumaɛ fiy :aloɛarabiyyat*. Edited by Sharaf, H. M. Dar Ɛa|lam Alkutub, Cairo, Egypt. 2nd Edition.

Ibnu Jinniy, A. U. (c.1913) *:alttas;oriyf :alomuluwkiy*. Edited by Alnaɛsaan, M. S. Published by Svarikat Alttamaddun Als;s;ina|ɛiyyat, Cairo, Egypt. Translated into English by Sakhnini, H. M. A. (1984) as: *Arabic Morphology as Described by Ibnu Jinniy in "At-Tasrif Al-Muluki"*. (Ph.D. Thesis) Indiana University. University Microfilms International, Ann Arbor, Michigan, USA.

Ibnu Mandhuwr, J. I. M. (1966) *lisa|n :aloɛarab*. Edited by Alkabiyr, A.; Hasaballah, M. A.; and Alsvaadliy, H. M. Dar Almaɛa|rif, Cairo, Egypt.

Kharma, N. (1979) *:ad;owa|: ɛalay :alddira|sa|t :allugawiyyat :alomuɛa|s;irat*. Mat;a|biɛ Dar Alqabas, Kuwait. 2nd Edition.

Majmaɛ Allugat Alɛarabiyyat. (1980) *:alomuɛojam :alowasiyt;*. Al:ida|rat Alɛa|mmat Lilmuɛjama|t Wa:ih-ya|: Alttura|t-. Dar Almaɛa|rif, Cairo, Egypt.

Niɛma, F. (1973) *mulaxxas; qawa|ɛid :allugat :aloɛarabiyyat. Volume I: qawa|ɛid :alnnah-ow. Volume II, qawa|ɛid :als;s;arof*. Mat;baɛat Nahd;at Misr, Cairo. Nasvr Almaktab Alɛilmiyy Liltta:liyf Walttarjamat, Cairo, Egypt.

Shariyf, M. A. A. (1979) *:alnnah-ow :aloja|miɛiyy, dira|sat tat;obiyqiyyat*. Dar Alzzahra|: Lit;t;iba|ɛat Walnnasvr, Egypt. 2nd Edition.

Shivtiel, A. (1989) 'tadoriys :allugat :aloɛarabiyyat ligayor :alnna|t;iqiyna biha|, masvoruwɛ ja|miɛat liydz, brit;a|niya|', in [1], 1987: 297–300.

Siybawayhi, A. I. U. (1970) *:alokita|b. Le Livre de Sîbawaihi, Traité de Grammaire Arabe*. Volumes I & II. Texte Arabe Publié par Derenbourg, H. Georg Olms Verlag, Hildesheim, New York. Printed in Germany. (Printed in Paris, 1881).

Tahhaan, R. (1972) *:alo:alosuniyyat :aloɛarabiyyat. Volume II: :alnnah-ow, :alojumolat, :alo-:usoluwb, :aloɤa|timat.* Dar Alkita|b Allubna|niyy, Beirut, Lebanon.

Yaɛquwb, E. B. (1986) *mawosuwɛat :alnnah-ow wa:als;s;arof wa:alo:iɛora|b.* Dar Alɛilm Lilmala|yiyn, Beirut, Lebanon. 1st Edition, February, 1986.

Zakariyya, M. (1982) *:alo:alosuniyyat :alttawoliydiyyat wa:alttah-owiyliyyat waqawa|ɛid :alługat :aloɛarabiyyat.* Almu:assasat Alja|miɛiyyat Lilddira|sa|t Walnnasvr Walttawziyɛ, Beirut, Lebanon. 1st Edition.

C NON-ARABIC WORKS

ACL (1985) *Computational Linguistics: Special Issues on Machine Translation, II.* XI (2–3), April–September, 1985.

ACL (1988) *Computational Linguistics: Special Issue on Tense and Aspect.* XIV (2), June, 1988. The MIT Press, Cambridge, Massachusetts.

ALPAC (1966) *Language and Machines, Computers in Translation and Linguistics.* Publication 1416, National Academy of Sciences, National research Council, Washington, D. C.

Abboud, V. C. (1971) *A Computer Assisted Instruction Program in the Arabic Writing System.* (Ph.D. Thesis) The University of Texas at Austin, Texas. University Microfilms Inc., Ann Arbor, Michigan, USA. Presented, 1970; Published; 1971.

Abd-Rabbo, M. Z. (1990) 'Sound Plural and Broken Plural Assignment in Classical Arabic', in Eid, Mushira., Ed. *Perspectives on Arabic Linguistics I. Papers from the First Annual Symposium on Arabic Linguistics*, (University of Utah, April 24–25, 1987). 1990: 55–93. Current Issues in Linguistic Theory 63. John Benjamins Publishing Company, Amsterdam/Philadelphia.

Albakkuwsh, A. (1970) 'Bibliographie Critique des Etudes Linguistiques Concernant la Tunisie', in *Revue Tunisienne des Sciences Sociales.* Publication du CERES, Tunis, Tunisia. 7ème Année (No. 20), Mars, 1970: 239–286.

Alfehri, A. A. (1982) *Linguistique Arabe: Forme et Interprétation.* Publications de la Faculté des Lettres et des Sciences Humaines de Rabat. Thèses et Mémoires No. 9.

Ali, N. (1989) 'Automatic Diacritization of Written Arabic', (sup. to) [2], 1989: Session 2A, 10 pp. (n.r.p.).

Alkhuwliy, M. A. (1979) *A Contrastive Transformational Grammar, Arabic and English.* E. J. Brill, The Netherlands.

Alkhuwliy, M. A. (1982) *A Dictionary of Theoretical Linguistics, English-Arabic.* Librairie du Liban, Beirut, Lebanon.

Allard, M. (1970) 'Notes sur l'Informatique au Service de la Langue Arabe', in *Actes Vème Congrès International d'Arabisants et d'Islamisants*, Bruxelles, 1970: 31–33.

Allerton, D. J. (1979) *Essentials of Grammatical Theory: A Consensus View of Syntax and Morphology.* Routledge and Kegan Paul, London, Boston, and Henley.

Anderson, S. R. (1982) 'Where's Morphology?', in *Linguistic Inquiry.* Volume XIII (4), Fall, 1982: 571–612. The MIT Press, Cambridge, Massachusetts.

Anderson, S. R. (1986) 'Disjunctive Ordering in Inflectional Morphology', in *Natural Language and Linguistic Theory.* Volume IV (1), February, 1986: 1–31. D. Reidel Publishing Company. Dordrecht, Holland/Boston, USA.

Anghelescu, N. (1974) 'Sur le Système de l'Article en Arabe', in *Revue Romaine de Linguistique.* Volume XIX (1), 1974: 45–52. Bucarest.

Aoun, J. (1979b) *On Government, Case-Marking, and Clitic Placement.* MIT Mimeograph.

Aristar, A. (1987) 'Unification and the Computational Analysis of Arabic', in *Computers and Translation.* II (2), April–June, 1987: 67–75. Paradigm Press, Inc.

Aronoff, M. (1976) *Word Formation in Generative Grammar.* Linguistic Inquiry Monograph One. The MIT Press, Cambridge, Massachusetts.

Bakalla, M. H. (1979) *The Morphological and Phonological Components of The Arabic Verb: (Meccan Arabic).* Longman and Librairie du Liban, Beirut, Lebanon.

Bakalla, M. H. (1983) *Arabic Linguistics: An Introduction and Bibliography.* Mansell Publishing Ltd, London. 1st Published, 1983; 2nd Revised Edition.

Bakir, M. J. (1980) *Aspects of Clause Structure in Arabic: A Study of Word Order Variation in Literary Arabic.* Indiana University Linguistics Club, Bloomington, Indiana.

Barr, A. and Feigenbaum, E. A. (1981) *The Handbook of Artificial Intelligence.* Volume I. William Kaufmann, Inc. Pitman Books Ltd, London, England. 1st Published in Great Britain, 1981.

Bathurst, R. D. (1971) 'Automatic Alphabetization of Arabic Words, a Problem of Graphic Morphology and Combinatorial Logic', in Wisbey, R. A., Ed. *The Computer in Literary and Linguistic Research, Papers from a Cambridge Symposium.* Volume I, 1971: 185–190. CUP, Cambridge.

Becker, J. D. (1984) 'Multilingual Word Processing', in *Scientific American.* CCLI (1), July, 1984: 82–93. New York, USA.

Becker, J. D. (1987) 'Arabic Word Processing', in *Communications of the ACM.* XXX (7), July, 1987: 600–616.

Becker-Makkai, V. (1970) 'Problems in the Computational Parsing of the Arabic Verb', in *Actes du Xème Congrès International des Linguistes*, Bucarest, 28 Août–2 Septembre, 1967: X (4), 927–931. Editions de la République Socialiste de Roumanie, Bucarest.

Beesley, K. (1990) 'Finite-State Description of Arabic Morphology', in [3], 1990: Session 6, 5 pp. (n.r.p.).

Beesley, K.; Buckwalter, T.; and Newton, S. (1989) 'Two-Level Finite State Analysis of Arabic Morphology', in [2], 1989: Session 2A, 10 pp. (n.r.p.).

Bejaoui, M. (1985) *Etude et Réalisation d'un Système Expert Appliqué à l'Analyse Morpho-Syntaxique de Phrases en Langue Arabe.* (Ph.D. Thesis) CERFIA. Toulouse, France.

Ben Hamadou, A. (1986a) 'Automatic Detection and Correction of Spelling Errors in Arabic Texts', in *Actes 2ème Conférence Internationale de Baghdad sur la Technologie de l'Ordinateur et ses Applications*, Baghdad, 24–26 Mars, 1986: 5 pp. (n.r.p.).

Ben Hamadou, A. (1986b) 'A Compression Technique for Arabic Dictionaries: the Affix Analysis', in [4], 1986: 286–288. ICCL.

Ben Hamadou, A. and Saidi, L. (1986) 'Analyse Morpho-Syntaxique de la Langue Arabe. Approche Système-Expert', in *8ème Séminaire Tuniso-Français en Informatique: Intelligence Artificielle.* Faculté des Sciences, Tunis, Mai, 1986: 19 pp. (n.r.p.).

Blachère, R. (1976) *Eléments de l'Arabe Classique.* Maisonneuve, G. P. and Larose, Editeurs. Paris. 4ème Edition, 1981.

Blau, J. (1973) 'Remarks on some Syntactic Trends in Modern Standard Arabic', in *Israel Oriental Studies.* III, 1973: 172–231, Tel Aviv University, Faculty of Humanities. The Central Press, Jerusalem.

Blau, J. (1976) 'Some Additional Observations on Syntactic Trends in Modern Standard Arabic', in *Israel Oriental Studies.* VI, 1976: 158–190, Tel Aviv University, Faculty of Humanities. The Central Press, Jerusalem.

Bloomfield, L. (1933) *Language.* George Allen and Unwin Ltd, London. 1st Published in Great Britain, 1935.

Bohas, G. (1975) 'Quelques Règles pour la Conjugaison du Verbe en Arabe Littéral', in l'Association de Professeurs de Langues Vivantes de l'Enseignement Publique. *Les Langues Modernes Revue et Bulletin*. LXIX (4), 1975: 345–356, Paris.

Borrmans, M. (1961) 'A Propos de l'Arabe Moderne. Notes Syntaxiques.', in *Revue de l'Institut des Belles Lettres Arabes*. XXIV, 1961: 363–372, Tunis.

Boubaker, O. (1986) *Al-Semss: Un Système Expert en Morphologie, Syntaxe et Sémantiques pour l'Etude de la Langue Arabe*. (Ph.D. Thesis) CERFIA. Toulouse, France.

Brockett, A. (1986) 'Arabic by Computer: Report on a Symposium Held at Leeds', (21–22 July, 1986) in *The CTISS File*. No. 2, November, 1986: 13. Published by SWURCC, University of Bath, England.

Brusset, J. and Abdelghani, S. (1989) 'Création d'une Base de Données Lexicale de l'Arabe Ecrit Utilisable par un Système Morpho-Syntaxique', in [1], 1987: 9–28 (French Section).

Cachia, P. (1973) *The Monitor, A Dictionary of Arabic Grammatical Terms: Arabic/English. English/Arabic*. Librairie du Liban, Beirut. Longman, London. Printed in Lebanon, Dar Alqalam Press Co.

Carbonell, J. G. and Tomita, M. (1987) 'Knowledge-Based Machine Translation, the CMU Approach', in Nirenburg, S., Ed. *Machine Translation: Theoretical and Methodological Issues*. 1987: 68–89. CUP, Cambridge.

Chaib, M. (1982) *Machine Translation and Arabic*. (M.A. Thesis) School of Modern Languages, University of Bath, England.

Charniak, E. and McDermott, D. (1985) *Introduction to Artificial Intelligence*. Addison Wesley Publishing Company, London, Amsterdam, Sydney.

Chomsky, N. (1957) *Syntactic Structures*. Mouton, The Hague. Janua Linguarum, Series Minor 4. 1st Published, 1957; 11th Printing, 1975; 13th Printing, 1978.

Chomsky, N. (1965) *Aspects of the Theory of Syntax*. The MIT Press, Cambridge, Massachusetts.

Chomsky, N. (1972b) 'Remarks on Nominalization', in Chomsky, N., Author. *Studies on Semantics in Generative Grammar*. 1972: 11–61. Mouton, The Hague. Janua Linguarum, Series Minor 107. 2nd Edition, 1975. 3rd Edition, 1980.

Chomsky, N. (1981) *Lectures on Government and Binding*. Studies in Generative Grammar, IX. Foris Publications, USA.

Chomsky, N. (1982) *Some Concepts and Consequences of the Theory of Government and Binding*. Linguistic Inquiry Monograph Six. The MIT Press, Cambridge, Massachusetts.

Cohen, D. (1961) 'Essai d'une Analyse Grammaticale de l'Arabe', in *La Traduction Automatique*. Volume II (3), 1961: 48–70.

Cohen, D. (1970) 'Remarques sur la Dérivation Nominale par Affixes dans Quelques Langues Sémitiques', in Cohen, D., Author. *Etudes de Linguistique Sémitique et Arabe*. 1970: 31–48. Mouton, The Hague. Janua Linguarum, Series Practica 81.

Colmerauer, A. (1978) 'Metamorphosis Grammars', in Bolc, L., Ed. *Natural Language Communication with Computers*. 1978: 133–139. Lecture Notes in Computer Science 63. Springer-Verlag, Berlin, Germany.

Commission Permanente de l'Arabe Fonctionnel, La. (1976) *L'Arabe Fonctionnel, Premier Cycle de l'Enseignement Primaire; Liste Français-Arabe*. Tunis. 1st Edition.

Crystal, D. (1971) *Linguistics*. Penguin Books, Middlesex, England.

Crystal, D. (1980) *A First Dictionary of Linguistics and Phonetics*. André Deutsch, London.

Czapkiewicz, A. (1965–1966) 'The Hypothetical Standard-Length of Words in Arabic and its Realization in the Vocabulary', (Documents et Communications) in *Folia Orientalia*

(KRAKÓW), 1965/1966: VII, 309–313.

Darcy, L. and Boston, L. (1983) *A Dictionary of Computer Terms.* Fontana Paperbacks. 1st Published in the USA, 1983. 1st Published by Fontana Paperbacks, 1984.

De Roeck, A. (1983) 'An Underview of Parsing', in King, M., Ed. *Parsing Natural Language.* 1983: 3–17. Academic Press, London, New York.

Debili, F. (1989) 'Analyse Morphologique: De l'Approche sans Dictionnaire à l'Approche sans Grammaire', (sup. to) [1], 1987: 12pp.

Debili, F. and Zouari, L. (1989) 'Analyse Morphologique de l'Arabe Ecrit Voyellé ou non Fondée sur la Construction Automatisée d'un Dictionnaire Arabe', (sup. to) [1], 1987: 7pp.

Degachi, A. (1984) *Word Orders of Nominal Sentences in M.S.A.* (M.A. Thesis) School of Modern Languages, University of Bath, England.

Degachi, A. and Smith, J. C. (1990) 'The Morphological Component of an Arabic Parser', in [3], 1990: Session 6, 9 pp. (n.r.p.).

Descout, R., Ed. (1983) *Applied Arabic Linguistics and Signal and Information Processing. Procs. of the Arab School on Science and Technology*, Rabat, Morocco, 26 September–5 October, 1983. Hemisphere Publishing Corporation, London. Distribution: Springer-Verlag.

Dey, P. (1986) 'Processing Word Order Variation within a Modified ID/LP Framework', in [4], 1986: 65–67. ICCL.

Di Sciullo, A. M. and Williams, E. (1987) *On the Definition of Word.* Linguistic Inquiry Monograph Fourteen. The MIT Press, Cambridge, Massachusetts.

Elsadany, T. A. and Hashish, M. A. (1989) 'An Arabic Morphological System', in *IBM Systems Journal.* Volume XXVIII (4), 1989: 600–612. IBM Co, USA.

Eltikaina, I. S. H. (1982) *A Lexical Approach to the Arabic Verb Conjugations.* (Ph.D. Thesis) The University of Washington. University Microfilms International, Ann Arbor, Michigan, USA.

Erickson, J. L. (1965) *English and Arabic: A Discussion of Contrastive Verbal Morphology.* (Ph.D. Thesis) The University of Texas at Austin, Texas. Xerox University Microfilms Inc., Ann Arbor, Michigan, USA, 1976.

Farghaly, A. and Brownfield, S. (1985) 'Towards a More Intelligent Software', in Saeed, A. M., Ed. 1985: Volume I, 27 pp. (n.r.p.).

Fenwick, P. (1987) 'Recent Developments in IT Standards of Interest to Translators', in Picken, C., Ed. *Translating and the Computer, VIII, A Profession on the Move*, 13–14 November, 1986: 31–39. ASLIB, London. 1st Published, 1987.

Finch, R. (1986) 'Prosodic Templates for the Sound Triliteral Verb of Arabic', in *Sophia Linguistica.* Volumes XX-XXI, 1986: 179–200. Japan.

Fitch, J. P. (1984) *LISP and Lambda Calculus: Lecture Notes, 1984.* Bath University, Bath. Unpublished.

Fitch, J. P. (1988) *Applications of Logic: Lecture Notes, 1988.* Bath University, Bath. Unpublished.

Fitch, J. P. and Norman, A. C. (1977) 'Implementing LISP in a High-Level Language', in *Software Practice and Experience* 7, 1977: 713–725.

Fitch, J. P. and Norman, A. C. (1985) *HLH Orion Cambridge LISP Manual.* Oxford, High-Level Hardware.

Fitzpatrick, E. and Sager, N. (1974) 'The Lexical Subclasses of the Linguistic String Parser', in *The American Journal of Computational Linguistics.* Microfiche II (1), 1974: 1–69. ACL.

Ford, N. (1987) *How Machines Think, a General Introduction to Artificial Intelligence Illus-*

trated in Prolog. John Wiley & Sons, New York.

Forsyth, R. and Rada, R. (1986) *Machine Learning, Applications in Expert Systems and Information Retrieval.* Ellis Horwood Limited, John Wiley & Sons, New York.

Frazier, W. M. (1976) *A Demonstration of a Computer Technique for Investigating Arabic Morphology.* Volumes I & II. (Ph.D. Thesis) The University of Michigan. Xerox University Microfilms International, Ann Arbor, Michigan, USA.

Gazdar, G. (1985) *Finite State Morphology: A Review of Koskenniemi (1983).* Report No. CSLI-85-32. CSLI, Leland Stanford Junior, Stanford, California.

Gelb, I. J. (1970) 'A Note on Morphographemics', in Cohen, D., Ed. *Mélanges Marcel Cohen, Etudes de Linguistique, Ethnographie et Sciences Connexes.* Mouton, The Hague. Janua Linguarum, Series Major 27.

Ghandour, Z. (1976) 'An Arabic Interface to APL', in *Informatics.* 1976: 483–492.

Gloess, P. Y. (1982) *Understanding LISP: A Concise Introduction to the Language of Artificial Intelligence.* Computer Series. An Alfred Handy Guide, Alfred Publishing Co, Inc., Sherman Oaks, California.

Greenberg, J. H. (1950) 'The Patterning of Root Morphemes in Semitic', in *Word.* VI, 1950: 162–181.

Greenberg, J. H. (1966) *Universals of Language, Report of a Conference Held at Dobbs Ferry,* New York, 13–15 April, 1961. The MIT Press, Cambridge, Massachusetts and London, England. 2nd Edition. 1st Published, 1963.

Grishman, R. (1976) 'A Survey of Syntactic Analysis Procedures for Natural Language', in *The American Journal of Computational Linguistics.* Microfiche XLVII (2), 1976: 1–97. ACL.

Hall, P. and Hussain, I. (1978) 'Design of Information systems for Arabic', in *Information Systems Methodology. Procs. of the Second Conference of the European Cooperation in Information,* Venice, Italy, 1978: 643–663. Published by Springer-Verlag, Berlin.

Hammond, M. (1988) 'Template Transfer in Arabic Broken Plurals', in *Natural Language and Linguistic Theory.* Volume VI (2), May, 1988: 247–270. D. Reidel Publishing Company. Dordrecht, Holland/Boston, USA. Kluwer Academic Publishers.

Hanks, P.; McLeod, W. T.; and Urdang, L.; Eds. (1986) *The Collins Dictionary of the English Language.* Collins, London and Glasgow. 2nd Edition. 1st Published, 1979. Reprinted, 1987 and 1988.

Hasemer, T. (1984) *A Beginner's Guide to LISP.* Addison-Wesley Publishers Ltd, Massachusetts, USA. Small Computer Series.

Hegazi, N. and Sharkawi, A. (1985) 'An Approach to a Computational Lexical Analyzer for Arabic Text', in *Procs. of the Workshop on Computer Processing and Transmission of the Arabic Language,* Kuwait, 1985.

Herdan, G. (1962a) *The Calculus of Linguistic Observations.* Mouton & Co. 'S-Gravenhage, The Hague. Janua Linguarum, Series Major 9.

Herdan, G. (1962b) 'The Patterning of Semitic Verbal Roots Subjected to Combinatory Analysis', in *Word.* XVIII (3), 1962: 262–268.

Hishmat, M. A. K. (1976) 'Vocalization and Computer Handling of Arabic Texts', in *Informatics.* 1976: 455–461.

Hlal, Y. (1984) 'Analyse Morphosyntaxique de l'Arabe non Voyellé', in *Bulletin d'Etudes Orientales.* Volume XXXIV, 1982: 59–85. Institut Français de Damas. Publié, Damas, 1984.

Hlal, Y. (1987) 'Information Systems and Arabic: The Use of Arabic in Information Systems', in Descout, R., Ed. 1983: 191–197.

Hudson, G. (1986) 'Arabic Root and Pattern Morphology without Tiers', in *Journal of Linguis-*

tics. Volume XXII, 1986: 85–122. Published for the Linguistics Society of Great Britain by CUP, Cambridge.

Hutchins, W. J. (1986) *Machine Translation: Past, Present, Future*. Ellis Horwood Limited, John Wiley & Sons, New York. 1st Published, 1986.

Hyder, S. S. and Ibrahim, K. (1980) 'Optimal Placement of Isolated Characters on Arabic Keyboards', in *Procs. of the International Symposium for Standardization of Codes, Character Sets and Keyboards for the Arabic Language in Computers*, Riyadh, Saudi Arabia, 1980: 140–180.

Hyder, S. S. and Richer, F. (1977) 'The Theory and Design of a System for Printing and Communication in Arabic-Farsi-Urdu languages', in *Bio-Sciences Communications*. III, 1977: 181–206. American Institute of Biological Sciences, Arlington, VA.

INTERGRAPH. (1987) *Clipper—32-Bit Microprocessor: Users' Manual*. Prentice-Hall, Inc., New Jersey, USA.

Ibrahim, M. M. A.; Clarke, J. D.; and Fahmy, A. A. (1989) 'Arabic in Machine Translation', in [2], 1989: Session 2A, 12 pp. (n.r.p.).

Jäppinen, H. and Ylilammi, M. (1986) 'Associative Model of Morphological Analysis: An Empirical Enquiry', in *Computational Linguistics*. XII (4), October–December, 1986: 257–272.

Jackendoff, R. (1977) *$\bar{X}$ Syntax: A Study of Phrase Structure*. Linguistic Inquiry Monograph Two. The MIT Press, Cambridge, Massachusetts.

Jensen, T. J. (1990) *Morphology. Word Structure in Generative Grammar*. Current Issues in Linguistic Theory 70. John Benjamins Publishing Company, Amsterdam/Philadelphia.

Jensen, T. J. and Stong-Jensen, M. (1984) 'Morphology is in the Lexicon', in *Linguistic Inquiry*. Volume XV (3), Summer, 1984: 474–498. The MIT Press, Cambridge, Massachusetts.

Johnson, R. (1983) 'Parsing with Transition Networks', in King, M., Ed. *Parsing Natural Language*. 1983: 59–72. Academic Press, London, New York.

Johnson, R. (1985) 'Parsing—an MT Perspective', in Jones, K. S. and Wilks, Y., Eds. *Automatic Natural Language Parsing*. 1985: 32–38. Ellis Horwood, England.

Killean, C. G. (1966) *The Deep Structure of the Noun Phrase in Modern Written Arabic*. (Ph.D. Thesis) The University of Michigan. Produced by Xerox University Microfilms International, Ann Arbor, Michigan, USA, 1977.

King, M. (1981) 'Design Characteristics of a Machine Translation system', in *Procs. of the Seventh IJCAI*, Vancouver, Canada. Volume I, 1981: 43–46.

Koskenniemi, K. M. (1983) *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production*. Department of General Linguistics, University of Helsinki. Publication No. 11.

Kowalski, R. A. (1980) *Logic for Problem Solving*. North Holland, New York, USA.

Lazrek, A. (1986) *Système Basé sur les Connaissances pour l'Analyse Morpho-Syntaxique Optimisée de la Langue Arabe*. (Ph.D. Thesis) CERFIA. Toulouse, France.

Lehmann, W. P. (1978) 'Machine Translation', in Belzer, J.; Holzman, A. G.; and Kent, A.; Eds. *Encyclopedia of Computer Science and Technology*. Volume X, 1978: 151–164. Dekker, New York and Basel.

Lewkowicz, N. M. K. (1967) *A Transformational Approach to the Syntax of Arabic Participles*. (Ph.D. Thesis) The University of Michigan. University Microfilms Inc., Ann Arbor, Michigan, USA.

Lewkowicz, N. M. K. (1971) 'Topic-Comment and Relative Clause in Arabic', in *Language*. Volume XLVII, 1971: 810–825.

Lyons, J. (1977) *Chomsky*. Fontana-collins, Glasgow, Great Britain. 1st Published, 1970. Revised Edition 1977. 12th Impression 1979.

MITRE. (1964) *English Preprocessor Manual*. Report SR-132. The MITRE Corporation, Bedford, Massachusetts.

Mackay, P. A. (1976) 'Computer Processing for Arabic Script Documents, Proposal for a Standardized Code', in Bergue, J. and Chevalier, D., Eds. *Les Arabes par leurs Archives, 15ème au 20ème Siècles*. 1976: Chapitre XXI, 257–271. CNRS, Paris.

Mackay, P. A. (1977) 'Setting Arabic with a Computer', in *Scholarly Publishing*. VIII, 1977: 142–150.

Mahadin, R. S. (1982) *The Morphophonemics of the Standard Arabic Triconsonantal Verbs*. (Ph.D. Thesis) The University of Pennsylvania. University Microfilms International, Ann Arbor, Michigan, USA.

Marti, J. (1984) *LISP: Lecture Notes, Chapter I*. 1984: 1–28. Bath University, Bath. Unpublished.

Masterman, M. (1979) 'The Essential Skills to be Acquired for Machine Translation', in Snell, B. M., Ed. 1978: 159–180.

Matthews, P. H. (1974) *Morphology: An Introduction to the Theory of Word Structure*. CUP, Cambridge.

McCarthy, J. (1981) 'A Prosodic Theory of Nonconcatenative Morphology', in *Linguistic Inquiry*. Volume XII (3), Summer, 1981: 373–418. The MIT Press, Cambridge, Massachusetts.

McCarthy, J. (1982) 'Prosodic Templates, Morphemic Templates, and Morphemic Tiers', in Van Der Hulst, H. and Smith, N., Eds. *The Structure of Phonological Representations (Part I)*. 1982: 191–223. Foris Publications, Dordrecht, Holland/Cinnaminson, USA.

McDermott, D. (1982) 'Artificial Intelligence Meets Natural Stupidity', in Haugeland, J., Ed. *Mind Design: Philosophy, Psychology, Artificial Intelligence*. 1982: 143–160. The MIT Press, Cambridge, Massachusetts and London, England. 1st MIT Edition, 1981.

Mehdi, S. A. (1985) *Arabic Language Parser*. Research Report No. R129. Department of Computer Science, University of Exeter, England. October, 1985.

Mehdi, S. A. (1986) *Arabic Language Semantic Analyzer*. Research Report No. R131. Department of Computer Science, University of Exeter, England. February, 1986.

Mourtaki, A. (1986) *A Propos de la Conception et l'Interprétation sur Micro-Ordinateur d'un Système Expert pour l'Analyse Morpho-Syntaxique d'une Phrase Arabe*. (Rapport Diplôme d'Etudes Approfondies) CERFIA. Toulouse, France.

Mryati, M. (1987) 'Statistical Studies of Arabic Language Roots', in Descout, R., Ed. 1983: 97–117.

Murray, J. A. H.; Bradley, H.; Craigie, W. A.; and Onions, C. T.; 1st Eds. (1989) *The Oxford English Dictionary*. Volume X. Oxford (Clarendon) University Press, Oxford. 2nd Edition prepared by Simpson, J. A. and Weiner, E. S. C. 2nd Edition combined with a Supplement edited by Burchfield, R. W.

Obermeier, K. K. (1987) 'Natural-Language Processing, an Introductory Look at Some of the Technology Used in this Area of Artificial Intelligence', in *Byte, The Small Systems Journal*. Volume XII (14), December, 1987: 225–232. A McGraw Hill Publication.

Pereira, F. C. N. (1985) 'A New Characterization of Attachment Preferences', in Dowty, D. R.; Karttunen, L.; and Zwicky, A. M.; Eds. *Natural-Language Parsing: Psychological, Computational, and Theoretical Perspectives*. 1985: 307–319. CUP, Cambridge.

Pereira, F. C. N. and Shieber, S. M. (1987) *Prolog and Natural-Language Analysis*. Lecture Notes Number 10. CSLI, Stanford, California.

Pereira, F. C. N. and Warren, D. H. D. (1980) 'Definite Clause Grammars for Language Analysis—A Survey of the Formalism and a Comparison with Augmented Transition Networks', in *Artificial Intelligence*. XIII, 1980: 231–278.

Peterson, D. (1972) 'Some Explanatory Methods of the Arab Grammarians', in Peranteau, P. M.; Levi, J. N.; and Phares, G. C.; Eds. *Papers from the Eighth Regional Meeting of the Chicago Linguistic Society*. 14–16 April, 1972: 504–515. Chicago Linguistic Society, Chicago, Illinois.

Petráček, K. (1960) 'A Study in the Structure of Arabic', in *Orientalia Pragensia I, Acta Universitatis Carolinae—Philologica*. I, 1960: 23–39, Praha.

Petrick, S. R. (1965) *A Recognition Procedure for Transformational Grammars*. (Ph.D. Thesis) Department of Modern Languages, MIT, Cambridge, Massachusetts.

Radford, A. (1981) *Transformational Syntax, A Student's Guide to Chomsky's Extended Standard Theory*. CUP, Cambridge, England.

Radford, A. (1988) *Transformational Grammar: A First Course*. CUP, Cambridge, England. 1st Published, 1988.

Ritchie, G. and Thompson, H. (1984) 'Natural Language Processing', in O'Shea, T. and Eisenstadt, M., Eds. *Artificial Intelligence: Tools, Techniques, and Applications*. 1984: 358–388. Harper and Row, Publishers, New York.

Roochnik, P. (1989) 'ARABATN: a Morphosyntactic Parser for Arabic', in [2], 1989: Session 2A, 13 pp. (n.r.p.).

Saad, G. N. (1982) *Transitivity, Causation and Passivization. A Semantic-Syntactic Study of the Verb in Classical Arabic*. Monograph No. 4. Kegan Paul International, London, Boston and Melbourne. 1st Published, 1982.

Saeed, A. M., Ed. (1985) *Proceedings of the First National Symposium on Language Teaching*. Language Centre, Kuwait University, Kuwait, 4–6 May, 1985: Volumes I & II.

Sager, J. C. (1979) 'Multilingual Communication: Chairman's Introductory Review of Translating and the Computer', in Snell, B. M., Ed. 1979: 1–25.

Sager, N. (1978) 'Natural Language Analysis and Processing', in Belzer, J.; Holzman, A. G.; and Kent, A.; Eds. *Encyclopedia of Computer Science and Technology*. Volume XI, 1978: 152–169. Dekker, New York and Basel.

Saidi, M. L. (1985) *Etude et Réalisation d'un Système Expert Appliqué à l'Analyse Morpho-Syntaxique de Phrase en Langue Arabe*. (Ph.D. Thesis) CERFIA. Toulouse, France.

Satterthwait, A. C. (1962) *Parallel Sentence Construction Grammars of Arabic and English*. Harvard University, Cambridge, Massachusetts.

Satterthwait, A. C. (1963) 'Computational Research in Arabic', in *Mechanical Translation*. II, 1963: 62–70. Cambridge, Massachusetts.

Satterthwait, A. C. (1965) 'Sentence-for-Sentence Translation: An Example', in *Mechanical Translation*. II, 1965: 14–38. Cambridge, Massachusetts.

Scalise, S. (1984) *Generative Morphology*. Studies in Generative Grammar, XVIII. Foris Publications, USA.

Schabert, P. (1973) 'Ein Automatisches Verfahren zur Morphologischen Analyse und zur Herstellung von Indices für Arabische Texte', in *Zeitschrift für die Alttestamentliche Wissenschaft*, Berlin, Volume CXXIII (2), 1973: 238–251.

Schmidt, R. W. (1971) *The Automatic Generation of Arabic Verb Forms from Consonantal Roots*. (M.A. Thesis) Brown University, USA.

Selkirk, E. O. (1982) *The Syntax of Words*. Linguistic Inquiry Monograph Seven. The MIT Press, Cambridge, Massachusetts.

Sieny, A. E. (1984) 'A Select Bibliography on Machine Translation and Machine-Aided Translation', in *Conference Procs. of the Saudi Arabian National Center for Science and Technology*, Riyadh, 1984: 201–210. Riyadh, Saudi Arabia.

Slocum, J. (1981a) *A Practical Comparison of Parsing Strategies for Machine Translation and Other Natural Language Processing Purposes.* (Ph.D. Thesis) The University of Texas at Austin, Texas. University Microfilms International, Ann Arbor, Michigan, USA.

Slocum, J. (1981b) 'A Practical Comparison of Parsing Strategies', in *Procs. of the 19th Annual Meeting of the ACL*, 29 June–1 July, 1981: 1–6. Stanford University, Stanford, California.

Slocum, J. (1985) 'A Survey of Machine Translation: Its History, Current Status, and Future Prospects', in *Computational Linguistics.* Volume XI (1), January–March, 1985: 1–17. ACL.

Smeaton, B. H. (1956) 'Some Problems in the Description of Arabic', in *Word.* Volume XII (3), December, 1956: 357–368.

Smith, J. C. (1989) 'L'Etat Actuel de la Traduction Automatique', in [1], 1987: 71–113 (French Section).

Snell, B. M., Ed. (1979) *Translating and the Computer, Procs. of a Seminar*, 14 November, 1978. North-Holland Publishing Co; Amsterdam; New York; Oxford.

Snow, J. (1965) *A Grammar of Modern Written Arabic Clauses.* (Ph.D. Thesis) The University of Michigan. University Microfilms Inc., Ann Arbor, Michigan, USA.

Stetkevych, J. (1970) *The Modern Arabic Literary Language: Lexical and Stylistic developments.* The University of Chicago Press, Chicago and London. Publications of the Center for Middle Eastern Studies, No. 6.

Tennant, H. (1981) *Natural Language Processing, an Introduction to an Emerging Technology.* PBI, a Petrocelli Book, New York/Princeton.

Thalouth, B. and Aldannan, A. (1985) 'A Comprehensive Arabic Morphological Analyser Generator', in *Procs. of Arab School on Science and Technology*, Syria, 1985.

Travis, D. A. (1978) *Inflectional Affixation in Transformational Grammar: Evidence from the Arabic Paradigm.* (Ph.D. Thesis) The University of North Carolina at Chapel Hill. University Microfilms International, Ann Arbor, Michigan, USA. Presented, 1977; Published, 1978.

Van Riet, S. (1970) 'Traductions Arabo-Latines et Informatique', in *Actes Vème Congrès International d'Etudes Arabes et Islamiques*, Bruxelles, 31 Août–6 Septembre, 1970: 473–487. Publié par le Centre pour l'Etude des Problèmes du Monde Musulman Contemporain, Bruxelles.

Vauquois, B. (1987) 'Automatic Computer Aided Translation and the Arabic Languages', in Descout, R., Ed. 1983: 171–190.

Wehr, H. (1980) *A Dictionary of Modern Written Arabic.* Edited by Cowan, J. M. Librairie du Liban, Beirut and Macdonald and Evans, London. 3rd Printing. Otto Harrassowitz, Wiesbaden; 1961, 1966, 1971, 1974.

Wilensky, R. (1984) *LISPcraft.* W. W. Norton and Company, New York; London.

Wilks, Y. (1979) 'Machine Translation and Artificial Intelligence', in Snell, B. M., Ed. 1978: 27–43.

Winograd, T. (1983) *Language as a Cognitive Process. Volume I: Syntax.* Addison-Wesley Publishing Company, Inc., London, England and California, USA.

Winograd, T. (1984) 'Computer Software for Working with Language', in *Scientific American.* Volume CCLI (3), September, 1984: 90–101. New York.

Winston, P. H. and Horn, B. K. P. (1981) *LISP.* Addison-Wesley Publishing Company. London, Amsterdam, Sydney.

Woods, W. (1970) 'Transition Network Grammars for Natural Language Analysis', in *Communications of the ACM*. XIII, No. 1, January, 1970: 591–606.

Woods, W. (1973) 'An Experimental Parsing System for Transition Network Grammars', in Rustin, R., Ed. *Natural Language Processing*. Courant Computer Science Symposium 8, 20–21 December, 1973: 111–154. Algorithmics Press, New York.

Wright, W. (1896–1898) *A Grammar of the Arabic Language*. Volumes I & II. CUP, Cambridge, England. Translated from the German of Caspari. Volume I, 1896. Volume II, 1898. 3rd Edition revised by Robertson, S. and Goeje, M. J.

Yip, M. (1988) 'Template Morphology and the Direction of Association', in *Natural Language and Linguistic Theory*. Volume VI (4), November, 1988: 551–577. D. Reidel Publishing Company. Dordrecht, Holland/Boston, USA. Kluwer Academic Publishers.

A MORPHOSYNTACTIC PROCESSOR OF MODERN STANDARD ARABIC

submitted by

Abdelmajid Degachi

for the degree of Ph.D.

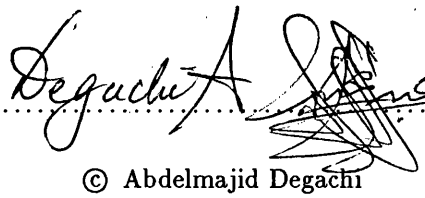
of the

University of Bath

December, 1990

School of Modern Languages
and International Studies

Signature of Author .....



VOLUME II
APPENDICES

Table of Contents

==0==<*****>==0==

List of Algorithms

==0==<*****>==0==

Table of Errata

Line	Page	Misprint	Correction
27	29	QHQ	PHQ
27	29	QCHQ	CPHQ
15	39	dfp Ø	dfp Ø Ø
20	39	dfp Ø	dfp Ø Ø
28	57	int	Ø
29	57	int	Ø
14	63	ii. IF LL the	ii. IF LL has the
32	68	MMK	the MMK
33	74	NBR & GDR	PRS & NBR
9	83	*STEP XII:*	*STEP XIII:*
31	84	ll	LL
41	84	ll	LL
28	86	ll	NN
29	86	LL	NN
6	89	ll	LL
12	89	ll	LL
11	92	;x	:x
14	124	0.002	0.02

Appendix A
SAMPLE ARABIC V-CONJUGATIONS, NOMINAL PATTERNING, AND EXAMPLES OF MORPHOLOGICAL CATEGORIES

A. SAMPLE ARABIC V-CONJUGATIONS

a. Basic Triliteral Sound Verbs

(1) Doubled; Faʕala-yafoʕulu: {d/nn, jdd, ʕmm}; e.g., 'd/nn', "think ...".

	PRACT	PRPAS
1.	d/ananotu	d/uninotu
2.	d/ananota	d/uninota
3.	d/ananoti	d/uninoti
4.	d/anna	d/unna
5.	d/annato	d/unnato
6.	d/ananna|	d/uninna|
7.	d/ananotuma|	d/uninotuma|
8.	d/anna|	d/unna|
8.	d/annata|	d/unnata|
9.	d/ananotumo	d/uninotumo
10.	d/ananotunna	d/uninotunna
11.	d/annuw|	d/unnuw|
12.	d/ananna	d/uninna

MPACT

	indicative	**subjunctive**	**jussive**
1.	:ad/unnu	:ad/unna	:ad/unna
2.	tad/unnu	tad/unna	tad/unna
3.	tad/unniyna	tad/unniy	tad/unniy
4.	yad/unnu	yad/unna	yad/unna
5.	tad/unnu	tad/unna	tad/unna
6.	nad/unnu	nad/unna	nad/unna
7.	tad/unna|ni	tad/unna|	tad/unna|
8.	yad/unna|ni	yad/unna|	yad/unna|
8.	tad/unna|ni	tad/unna|	tad/unna|
9.	tad/unnuwna	tad/unnuw|	tad/unnuw|
10.	tad/onunna	tad/onunna	tad/onunna
11.	yad/unnuwna	yad/unnuw|	yad/unnuw|
12.	yad/onunna	yad/onunna	yad/onunna

MPPAS

	indicative	**subjunctive**	**jussive**
1.	:ud/annu	:ud/anna	:ud/anna
2.	tud/annu	tud/anna	tud/anna
3.	tud/anniyna	tud/anniy	tud/anniy
4.	yud/annu	yud/anna	yud/anna
5.	tud/annu	tud/anna	tud/anna
6.	nud/annu	nud/anna	nud/anna
7.	tud/anna|ni	tud/anna|	tud/anna|
8.	yud/anna|ni	yud/anna|	yud/anna|
8.	tud/anna|ni	tud/anna|	tud/anna|
9.	tud/annuwna	tud/annuw|	tud/annuw|
10.	tud/onanna	tud/onanna	tud/onanna
11.	yud/annuwna	yud/annuw|	yud/annuw|
12.	yud/onanna	yud/onanna	yud/onanna

(2) Doubled; Faʕala-yafoʕalu: {:mm}

	PRACT	PRPAS
1.	:amamotu	N/A.
2.	:amamota	
3.	:amamoti	
4.	:amma	
5.	:ammato	
6.	:amamona|	
7.	:amamotuma|	
8.	:amma|	
8.	:ammata|	
9.	:amamotumo	
10.	:amamotunna	
11.	:ammuw|	
12.	:amamona	

MPACT

	indicative	**subjunctive**	**jussive**
1.	:a:ammu	:a:amma	:a:amma
2.	ta:ammu	ta:amma	ta:amma
3.	ta:ammiyna	ta:ammiy	ta:ammiy
4.	ya:ammu	ya:amma	ya:amma
5.	ta:ammu	ta:amma	ta:amma
6.	na:ammu	na:amma	na:amma
7.	ta:amma|ni	ta:amma|	ta:amma|
8.	ya:amma|ni	ya:amma|	ya:amma|
8.	ta:amma|ni	ta:amma|	ta:amma|
9.	ta:ammuwna	ta:ammuw|	ta:ammuw|
10.	ta:omamona	ta:omamona	ta:omamona
11.	ya:ammuwna	ya:ammuw|	ya:ammuw|
12.	ya:omamona	ya:omamona	ya:omamona

MPPAS

N/A.

(3) Solid; Faɛila-yafoɛilu: {h-sb}

	PRACT	PRPAS
1.	h-asibotu	h-usibotu
2.	h-asibota	h-usibota
3.	h-asiboti	h-usiboti
4.	h-asiba	h-usiba
5.	h-asibato	h-usibato
6.	h-asibona|	h-usibona|
7.	h-asibotuma|	h-usibotuma|
8.	h-asiba|	h-usiba|
8.	h-asibata|	h-usibata|
9.	h-asibotumo	h-usibotumo
10.	h-asibotunna	h-usibotunna
11.	h-asibuw|	h-usibuw|
12.	h-asibona	h-usibona

MPACT

	indicative	**subjunctive**	**jussive**
1.	:ah-osibu	:ah-osiba	:ah-osibo
2.	tah-osibu	tah-osiba	tah-osibo
3.	tah-osibiyna	tah-osibiy	tah-osibiy
4.	yah-osibu	yah-osiba	yah-osibo
5.	tah-osibu	tah-osiba	tah-osibo
6.	nah-osibu	nah-osiba	nah-osibo
7.	tah-osiba|ni	tah-osiba|	tah-osiba|
8.	yah-osiba|ni	yah-osiba|	yah-osiba|
8.	tah-osiba|ni	tah-osiba|	tah-osiba|
9.	tah-osibuwna	tah-osibuw|	tah-osibuw|
10.	tah-osibona	tah-osibona	tah-osibona
11.	yah-osibuwna	yah-osibuw|	yah-osibuw|
12.	yah-osibona	yah-osibona	yah-osibona

MPPAS

	indicative	**subjunctive**	**jussive**
1.	:uh-osabu	:uh-osaba	:uh-osabo
2.	tuh-osabu	tuh-osaba	tuh-osabo
3.	tuh-osabiyna	tuh-osabiy	tuh-osabiy
4.	yuh-osabu	yuh-osaba	yuh-osabo
5.	tuh-osabu	tuh-osaba	tuh-osabo
6.	nuh-osabu	nuh-osaba	nuh-osabo
7.	tuh-osaba|ni	tuh-osaba|	tuh-osaba|
8.	yuh-osaba|ni	yuh-osaba|	yuh-osaba|
8.	tuh-osaba|ni	tuh-osaba|	tuh-osaba|
9.	tuh-osabuwna	tuh-osabuw|	tuh-osabuw|
10.	tuh-osabona	tuh-osabona	tuh-osabona
11.	yuh-osabuwna	yuh-osabuw|	yuh-osabuw|
12.	yuh-osabona	yuh-osabona	yuh-osabona

(4) Solid; Faɛula-yafoɛulu: {h-sn, krm, kbr, kt-r, s;gr, qs;r, jml, fqr, t-mn, svrf}; e.g., 'h-sn', "to become good ...".

PRACT	PRPAS
1. h-asunotu	N/A.
2. h-asunota	
3. h-asunoti	
4. h-asuna	
5. h-asunato	
6. h-asunna|	
7. h-asunotuma|	
8. h-asuna|	
8. h-asunata|	
9. h-asunotumo	
10. h-asunotunna	
11. h-asunuw|	
12. h-asunna	

MPACT

	indicative	**subjunctive**	**jussive**
1.	:ah-osunu	:ah-osuna	:ah-osuno
2.	tah-osunu	tah-osuna	tah-osuno
3.	tah-osuniyna	tah-osuniy	tah-osuniy
4.	yah-osunu	yah-osuna	yah-osuno
5.	tah-osunu	tah-osuna	tah-osuno
6.	nah-osunu	nah-osuna	nah-osuno
7.	tah-osuna|ni	tah-osuna|	tah-osuna|
8.	yah-osuna|ni	yah-osuna|	yah-osuna|
8.	tah-osuna|ni	tah-osuna|	tah-osuna|
9.	tah-osunuwna	tah-osunuw|	tah-osunuw|
10.	tah-osunna	tah-osunna	tah-osunna
11.	yah-osunuwna	yah-osunuw|	yah-osunuw|
12.	yah-osunna	yah-osunna	yah-osunna

MPPAS
N/A.

(5) Solid; Faɛala-yafoɛulu: {ktb, svkr, h-sb, drs, frsv, qtl, rbɛ, skn, s;dq, nfd-, jbl, h-mr, zrq}; e.g., 'ktb', "to write".

	PRACT	PRPAS
1.	katabotu	kutibotu
2.	katabota	kutibota
3.	kataboti	kutiboti
4.	kataba	kutiba
5.	katabato	kutibato
6.	katabona|	kutibona|
7.	katabotuma|	kutibotuma|
8.	kataba|	kutiba|
8.	katabata|	kutibata|
9.	katabotumo	kutibotumo
10.	katabotunna	kutibotunna
11.	katabuw|	kutibuw|
12.	katabona	kutibona

MPACT

	indicative	**subjunctive**	**jussive**
1.	:akotubu	:akotuba	:akotubo
2.	takotubu	takotuba	takotubo
3.	takotubiyna	takotubiy	takotubiy
4.	yakotubu	yakotuba	yakotubo
5.	takotubu	takotuba	takotubo
6.	nakotubu	nakotuba	nakotubo
7.	takotuba|ni	takotuba|	takotuba|
8.	yakotuba|ni	yakotuba|	yakotuba|
8.	takotuba|ni	takotuba|	takotuba|
9.	takotubuwna	takotubuw|	takotubuw|
10.	takotubona	takotubona	takotubona
11.	yakotubuwna	yakotubuw|	yakotubuw|
12.	yakotubona	yakotubona	yakotubona

MPPAS

	indicative	**subjunctive**	**jussive**
1.	:ukotabu	:ukotaba	:ukotabo
2.	tukotabu	tukotaba	tukotabo
3.	tukotabiyna	tukotabiy	tukotabiy
4.	yukotabu	yukotaba	yukotabo
5.	tukotabu	tukotaba	tukotabo
6.	nukotabu	nukotaba	nukotabo
7.	tukotaba|ni	tukotaba|	tukotaba|
8.	yukotaba|ni	yukotaba|	yukotaba|
8.	tukotaba|ni	tukotaba|	tukotaba|
9.	tukotabuwna	tukotabuw|	tukotabuw|
10.	tukotabona	tukotabona	tukotabona
11.	yukotabuwna	yukotabuw|	yukotabuw|
12.	yukotabona	yukotabona	yukotabona

(6) Solid; Faɛala-yafoɛalu: {fth-, mnh-, mnɛ, d-hb, xd;r}; e.g., 'fth-', "to open ...".

	PRACT	PRPAS
1.	fatah-otu	futih-otu
2.	fatah-ota	futih-ota
3.	fatah-oti	futih-oti
4.	fatah-a	futih-a
5.	fatah-ato	futih-ato
6.	fatah-ona|	futih-ona|
7.	fatah-otuma|	futih-otuma|
8.	fatah-a|	futih-a|
8.	fatah-ata|	futih-ata|
9.	fatah-otumo	futih-otumo
10.	fatah-otunna	futih-otunna
11.	fatah-uw|	futih-uw|
12.	fatah-ona	futih-ona

MPACT

	indicative	**subjunctive**	**jussive**
1.	:afotah-u	:afotah-a	:afotah-o
2.	tafotah-u	tafotah-a	tafotah-o
3.	tafotah-iyna	tafotah-iy	tafotah-iy
4.	yafotah-u	yafotah-a	yafotah-o
5.	tafotah-u	tafotah-a	tafotah-o
6.	nafotah-u	nafotah-a	nafotah-o
7.	tafotah-a|ni	tafotah-a|	tafotah-a|
8.	yafotah-a|ni	yafotah-a|	yafotah-a|
8.	tafotah-a|ni	tafotah-a|	tafotah-a|
9.	tafotah-uwna	tafotah-uw|	tafotah-uw|
10.	tafotah-ona	tafotah-ona	tafotah-ona
11.	yafotah-uwna	yafotah-uw|	yafotah-uw|
12.	yafotah-ona	yafotah-ona	yafotah-ona

MPPAS

	indicative	**subjunctive**	**jussive**
1.	:ufotah-u	:ufotah-a	:ufotah-o
2.	tufotah-u	tufotah-a	tufotah-o
3.	tufotah-iyna	tufotah-iy	tufotah-iy
4.	yufotah-u	yufotah-a	yufotah-o
5.	tufotah-u	tufotah-a	tufotah-o
6.	nufotah-u	nufotah-a	nufotah-o
7.	tufotah-a|ni	tufotah-a|	tufotah-a|
8.	yufotah-a|ni	yufotah-a|	yufotah-a|
8.	tufotah-a|ni	tufotah-a|	tufotah-a|
9.	tufotah-uwna	tufotah-uw|	tufotah-uw|
10.	tufotah-ona	tufotah-ona	tufotah-ona
11.	yufotah-uwna	yufotah-uw|	yufotah-uw|
12.	yufotah-ona	yufotah-ona	yufotah-ona

(7) Solid/'Mahomuwz'; Faɛala-yafoɛilu: {jls, ksr, d;rb, nzl, h-ml, qlm, svbk, ɛsvr, sds, sbɛ, t-lt-, t-mn, tsɛ, xms, :lf, s;fr, h-dq}; e.g., 'ksr', "to break".

	PRACT	PRPAS
1.	kasarotu	kusirotu
2.	kasarota	kusirota
3.	kasaroti	kusiroti
4.	kasara	kusira
5.	kasarato	kusirato
6.	kasarona|	kusirona|
7.	kasarotuma|	kusirotuma|
8.	kasara|	kusira|
8.	kasarata|	kusirata|
9.	kasarotumo	kusirotumo
10.	kasarotunna	kusirotunna
11.	kasaruw|	kusiruw|
12.	kasarona	kusirona

MPACT

	indicative	**subjunctive**	**jussive**
1.	:akosiru	:akosira	:akosiro
2.	takosiru	takosira	takosiro
3.	takosiriyna	takosiriy	takosiriy
4.	yakosiru	yakosira	yakosiro
5.	takosiru	takosira	takosiro
6.	nakosiru	nakosira	nakosiro
7.	takosira|ni	takosira|	takosira|
8.	yakosira|ni	yakosira|	yakosira|
8.	takosira|ni	takosira|	takosira|
9.	takosiruwna	takosiruw|	takosiruw|
10.	takosirona	takosirona	takosirona
11.	yakosiruwna	yakosiruw|	yakosiruw|
12.	yakosirona	yakosirona	yakosirona

MPPAS

	indicative	**subjunctive**	**jussive**
1.	:ukosaru	:ukosara	:ukosaro
2.	tukosaru	tukosara	tukosaro
3.	tukosariyna	tukosariy	tukosariy
4.	yukosaru	yukosara	yukosaro
5.	tukosaru	tukosara	tukosaro
6.	nukosaru	nukosara	nukosaro
7.	tukosara|ni	tukosara|	tukosara|
8.	yukosara|ni	yukosara|	yukosara|
8.	tukosara|ni	tukosara|	tukosara|
9.	tukosaruwna	tukosaruw|	tukosaruw|
10.	tukosarona	tukosarona	tukosarona
11.	yukosaruwna	yukosaruw|	yukosaruw|
12.	yukosarona	yukosarona	yukosarona

(8) Solid/'Mahomuwz'; Faɛila-yafoɛalu: {fhm, ɛlm, qdm, svrb, rkb, mr:, sɛd, rjl}; e.g., 'fhm', "to understand".

	PRACT	PRPAS
1.	fahimotu	fuhimotu
2.	fahimota	fuhimota
3.	fahimoti	fuhimoti
4.	fahima	fuhima
5.	fahimato	fuhimato
6.	fahimona|	fuhimona|
7.	fahimotuma|	fuhimotuma|
8.	fahima|	fuhima|
8.	fahimata|	fuhimata|
9.	fahimotumo	fuhimotumo
10.	fahimotunna	fuhimotunna
11.	fahimuw|	fuhimuw|
12.	fahimona	fuhimona

MPACT

	indicative	**subjunctive**	**jussive**
1.	:afohamu	:afohama	:afohamo
2.	tafohamu	tafohama	tafohamo
3.	tafohamiyna	tafohamiy	tafohamiy
4.	yafohamu	yafohama	yafohamo
5.	tafohamu	tafohama	tafohamo
6.	nafohamu	nafohama	nafohamo
7.	tafohama|ni	tafohama|	tafohama|
8.	yafohama|ni	yafohama|	yafohama|
8.	tafohama|ni	tafohama|	tafohama|
9.	tafohamuwna	tafohamuw|	tafohamuw|
10.	tafohamona	tafohamona	tafohamona
11.	yafohamuwna	yafohamuw|	yafohamuw|
12.	yafohamona	yafohamona	yafohamona

MPPAS

	indicative	**subjunctive**	**jussive**
1.	:ufohamu	:ufohama	:ufohamo
2.	tufohamu	tufohama	tufohamo
3.	tufohamiyna	tufohamiy	tufohamiy
4.	yufohamu	yufohama	yufohamo
5.	tufohamu	tufohama	tufohamo
6.	nufohamu	nufohama	nufohamo
7.	tufohama|ni	tufohama|	tufohama|
8.	yufohama|ni	yufohama|	yufohama|
8.	tufohama|ni	tufohama|	tufohama|
9.	tufohamuwna	tufohamuw|	tufohamuw|
10.	tufohamona	tufohamona	tufohamona
11.	yufohamuwna	yufohamuw|	yufohamuw|
12.	yufohamona	yufohamona	yufohamona

b. Augmented Triliteral Sound Verbs

(1) Faɛɛala-yufaɛɛilu: {kttb, drrs, frrsv, svrrb, jlls, nzzl, fhhm, ɛllm, h-ssn, krrm, kbbr, kt-t-r, s;ggr, qs;s;r, jmml, fqqr, jddd, t-mmn, svrrf, h-mml, s;ffr, rkkb, mrr:, rjjl, xd;d;r, :mmm, ɛmmm, ftth-, kssr, qttl, skkn, d;rrb, rbbɛ, qllm, svbbk, ɛsvsvr, sdds, sbbɛ, t-llt-, xmms, :llf, s;ddq, nffd-, h-ddq, jbbl, h-mmr}; e.g., 'drrs', "to teach".

	PRACT	PRPAS
1.	darrasotu	durrisotu
2.	darrasota	durrisota
3.	darrasoti	durrisoti
4.	darrasa	durrisa
5.	darrasato	durrisato
6.	darrasona|	durrisona|
7.	darrasotuma|	durrisotuma|
8.	darrasa|	durrisa|
8.	darrasata|	durrisata|
9.	darrasotumo	durrisotumo
10.	darrasotunna	durrisotunna
11.	darrasuw|	durrisuw|
12.	darrasona	durrisona

MPACT

	indicative	**subjunctive**	**jussive**
1.	:udarrisu	:udarrisa	:udarriso
2.	tudarrisu	tudarrisa	tudarriso
3.	tudarrisiyna	tudarrisiy	tudarrisiy
4.	yudarrisu	yudarrisa	yudarriso
5.	tudarrisu	tudarrisa	tudarriso
6.	nudarrisu	nudarrisa	nudarriso
7.	tudarrisa|ni	tudarrisa|	tudarrisa|
8.	yudarrisa|ni	yudarrisa|	yudarrisa|
8.	tudarrisa|ni	tudarrisa|	tudarrisa|
9.	tudarrisuwna	tudarrisuw|	tudarrisuw|
10.	tudarrisona	tudarrisona	tudarrisona
11.	yudarrisuwna	yudarrisuw|	yudarrisuw|
12.	yudarrisona	yudarrisona	yudarrisona

MPPAS

	indicative	**subjunctive**	**jussive**
1.	:udarrasu	:udarrasa	:udarraso
2.	tudarrasu	tudarrasa	tudarraso
3.	tudarrasiyna	tudarrasiy	tudarrasiy
4.	yudarrasu	yudarrasa	yudarraso
5.	tudarrasu	tudarrasa	tudarraso
6.	nudarrasu	nudarrasa	nudarraso
7.	tudarrasa|ni	tudarrasa|	tudarrasa|
8.	yudarrasa|ni	yudarrasa|	yudarrasa|
8.	tudarrasa|ni	tudarrasa|	tudarrasa|
9.	tudarrasuwna	tudarrasuw|	tudarrasuw|
10.	tudarrasona	tudarrasona	tudarrasona
11.	yudarrasuwna	yudarrasuw|	yudarrasuw|
12.	yudarrasona	yudarrasona	yudarrasona

c. Basic Triliteral Defective Verbs

(1) Quasi-Sound; Faɛala-yafoɛilu: {ld, rd, h-d}; e.g., 'ld', "to give birth".

	PRACT	PRPAS
1.	waladotu	wulidotu
2.	waladota	wulidota
3.	waladoti	wulidoti
4.	walada	wulida
5.	waladato	wulidato
6.	waladona|	wulidona|
7.	waladotuma|	wulidotuma|
8.	walada|	wulida|
8.	waladata|	wulidata|
9.	waladotumo	wulidotumo
10.	waladotunna	wulidotunna
11.	waladuw|	wuliduw|
12.	waladona	wulidona

MPACT

	indicative	**subjunctive**	**jussive**
1.	:alidu	:alida	:alido
2.	talidu	talida	talido
3.	talidiyna	talidiy	talidiy
4.	yalidu	yalida	yalido
5.	talidu	talida	talido
6.	nalidu	nalida	nalido
7.	talida|ni	talida|	talida|
8.	yalida|ni	yalida|	yalida|
8.	talida|ni	talida|	talida|
9.	taliduwna	taliduw|	taliduw|
10.	talidona	talidona	talidona
11.	yaliduwna	yaliduw|	yaliduw|
12.	yalidona	yalidona	yalidona

MPPAS

	indicative	**subjunctive**	**jussive**
1.	:uwladu	:uwlada	:uwlado
2.	tuwladu	tuwlada	tuwlado
3.	tuwladiyna	tuwladiy	tuwladiy
4.	yuwladu	yuwlada	yuwlado
5.	tuwladu	tuwlada	tuwlado
6.	nuwladu	nuwlada	nuwlado
7.	tuwlada|ni	tuwlada|	tuwlada|
8.	yuwlada|ni	yuwlada|	yuwlada|
8.	tuwlada|ni	tuwlada|	tuwlada|
9.	tuwladuwna	tuwladuw|	tuwladuw|
10.	tuwladona	tuwladona	tuwladona
11.	yuwladuwna	yuwladuw|	yuwladuw|
12.	yuwladona	yuwladona	yuwladona

(2) Quasi-Sound; Faɛula-yafoɛulu: {sm}

	PRACT	PRPAS
1.	wasumotu	N/A.
2.	wasumota	
3.	wasumoti	
4.	wasuma	
5.	wasumato	
6.	wasumona|	
7.	wasumotuma|	
8.	wasuma|	
8.	wasumata|	
9.	wasumotumo	
10.	wasumotunna	
11.	wasumuw|	
12.	wasumona	

MPACT

	indicative	**subjunctive**	**jussive**
1.	:awosumu	:awosuma	:awosumo
2.	tawosumu	tawosuma	tawosumo
3.	tawosumiyna	tawosumiy	tawosumiy
4.	yawosumu	yawosuma	yawosumo
5.	tawosumu	tawosuma	tawosumo
6.	nawosumu	nawosuma	nawosumo
7.	tawosuma|ni	tawosuma|	tawosuma|
8.	yawosuma|ni	yawosuma|	yawosuma|
8.	tawosuma|ni	tawosuma|	tawosuma|
9.	tawosumuwna	tawosumuw|	tawosumuw|
10.	tawosumona	tawosumona	tawosumona
11.	yawosumuwna	yawosumuw|	yawosumuw|
12.	yawosumona	yawosumona	yawosumona

MPPAS
N/A.

(3) Hollow; Faɛila: {ls}

	PRACT	PRPAS
1.	lasotu	N/A.
2.	lasota	
3.	lasoti	
4.	layosa	
5.	layosato	
6.	lasona|	
7.	lasotuma|	
8.	layosa|	
8.	layosata|	
9.	lasotumo	
10.	lasotunna	
11.	layosuw|	
12.	lasona	

MPACT, MPPAS
N/A.

(4) Hollow; Faɛila-yafoɛalu: {:l}

	PRACT	PRPAS
1.	:awilotu	N/A.
2.	:awilota	
3.	:awiloti	
4.	:awila	
5.	:awilato	
6.	:awilona|	
7.	:awilotuma|	
8.	:awila|	
8.	:awilata|	
9.	:awilotumo	
10.	:awilotunna	
11.	:awiluw|	
12.	:awilona	

MPACT

	indicative	**subjunctive**	**jussive**
1.	:a:owalu	:a:owala	:a:owalo
2.	ta:owalu	ta:owala	ta:owalo
3.	ta:owaliyna	ta:owaliy	ta:owaliy
4.	ya:owalu	ya:owala	ya:owalo
5.	ta:owalu	ta:owala	ta:owalo
6.	na:owalu	na:owala	na:owalo
7.	ta:owala|ni	ta:owala|	ta:owala|
8.	ya:owala|ni	ya:owala|	ya:owala|
8.	ta:owala|ni	ta:owala|	ta:owala|
9.	ta:owaluwna	ta:owaluw|	ta:owaluw|
10.	ta:owalona	ta:owalona	ta:owalona
11.	ya:owaluwna	ya:owaluw|	ya:owaluw|
12.	ya:owalona	ya:owalona	ya:owalona

MPPAS
N/A.

(5) Hollow; Faɛala-yafoɛulu: {kn, qm, t;l, bb, jd, xl, s:, sd}; e.g., 'kn', "to be".

	PRACT	PRPAS
1.	kunotu	N/A.
2.	kunota	
3.	kunoti	
4.	ka|na	
5.	ka|nato	
6.	kunna|	
7.	kunotuma|	
8.	ka|na|	
8.	ka|nata|	
9.	kunotumo	
10.	kunotunna	
11.	ka|nuw|	
12.	kunna	

MPACT

	indicative	**subjunctive**	**jussive**
1.	:akuwnu	:akuwna	:akuno
2.	takuwnu	takuwna	takuno
3.	takuwniyna	takuwniy	takuwniy
4.	yakuwnu	yakuwna	yakuno
5.	takuwnu	takuwna	takuno
6.	nakuwnu	nakuwna	nakuno
7.	takuwna|ni	takuwna|	takuwna|
8.	yakuwna|ni	yakuwna|	yakuwna|
8.	takuwna|ni	takuwna|	takuwna|
9.	takuwnuwna	takuwnuw|	takuwnuw|
10.	takunna	takunna	takunna
11.	yakuwnuwna	yakuwnuw|	yakuwnuw|
12.	yakunna	yakunna	yakunna

MPPAS

N/A.

(6) Hollow; Faɛala-yafoɛilu: {s;r, bt, bd;}; e.g., 's;r', "to become".

PRACT

1. s;irotu
2. s;irota
3. s;iroti
4. s;a\|ra
5. s;a\|rato
6. s;irona\|
7. s;irotuma\|
8. s;a\|ra\|
8. s;a\|rata\|
9. s;irotumo
10. s;irotunna
11. s;a\|ruw\|
12. s;irona

PRPAS

N/A.

MPACT

	indicative	**subjunctive**	**jussive**
1.	:as;iyru	:as;iyra	:as;iro
2.	tas;iyru	tas;iyra	tas;iro
3.	tas;iyriyna	tas;iyriy	tas;iyriy
4.	yas;iyru	yas;iyra	yas;iro
5.	tas;iyru	tas;iyra	tas;iro
6.	nas;iyru	nas;iyra	nas;iro
7.	tas;iyra|ni	tas;iyra|	tas;iyra|
8.	yas;iyra|ni	yas;iyra|	yas;iyra|
8.	tas;iyra|ni	tas;iyra|	tas;iyra|
9.	tas;iyruwna	tas;iyruw|	tas;iyruw|
10.	tas;irona	tas;irona	tas;irona
11.	yas;iyruwna	yas;iyruw|	yas;iyruw|
12.	yas;irona	yas;irona	yas;irona

MPPAS

N/A.

(7) Weak; Faɛala-yafoɛilu: {bn, t-n}; e.g., 'bn', "to build".

	PRACT	PRPAS
1.	banayotu	buniyotu
2.	banayota	buniyota
3.	banayoti	buniyoti
4.	banay	buniya
5.	banato	buniyato
6.	banayona|	buniyona|
7.	banayotuma|	buniyotuma|
8.	banaya|	buniya|
8.	banata|	buniyata|
9.	banayotumo	buniyotumo
10.	banayotunna	buniyotunna
11.	banawo|	bunuw|
12.	banayona	buniyona

MPACT

	indicative	**subjunctive**	**jussive**
1.	:aboniy	:aboniya	:aboni
2.	taboniy	taboniya	taboni
3.	taboniyna	taboniy	taboniy
4.	yaboniy	yaboniya	yaboni
5.	taboniy	taboniya	taboni
6.	naboniy	naboniya	naboni
7.	taboniya|ni	taboniya|	taboniya|
8.	yaboniya|ni	yaboniya|	yaboniya|
8.	taboniya|ni	taboniya|	taboniya|
9.	tabonuwna	tabonuw|	tabonuw|
10.	taboniyona	taboniyona	taboniyona
11.	yabonuwna	yabonuw|	yabonuw|
12.	yaboniyona	yaboniyona	yaboniyona

MPPAS

	indicative	**subjunctive**	**jussive**
1.	:ubonay	:ubonay	:ubona
2.	tubonay	tubonay	tubona
3.	tubonayona	tubonayo	tubonayo
4.	yubonay	yubonay	yubona
5.	tubonay	tubonay	tubona
6.	nubonay	nubonay	nubona
7.	tubonaya|ni	tubonaya|	tubonaya|
8.	yubonaya|ni	yubonaya|	yubonaya|
8.	tubonaya|ni	tubonaya|	tubonaya|
9.	tubonawona	tubonawo|	tubonawo|
10.	tubonayona	tubonayona	tubonayona
11.	yubonawona	yubonawo|	yubonawo|
12.	yubonayona	yubonayona	yubonayona

(8) Weak; Faɛala-yafoɛalu: {d-k, m:}; e.g., 'm:', "to make 100 fold".

	PRACT	PRPAS
1.	ma:ayotu	mu:iyotu
2.	ma:ayota	mu:iyota
3.	ma:ayoti	mu:iyoti
4.	ma:ay	mu:iya
5.	ma:ato	mu:iyato
6.	ma:ayona|	mu:iyona|
7.	ma:ayotuma|	mu:iyotuma|
8.	ma:aya|	mu:iya|
8.	ma:ata|	mu:iyata|
9.	ma:ayotumo	mu:iyotumo
10.	ma:ayotunna	mu:iyotunna
11.	ma:awo|	mu:uw|
12.	ma:ayona	mu:iyona

MPACT

	indicative	**subjunctive**	**jussive**
1.	:amo:ay	:amo:ay	:amo:a
2.	tamo:ay	tamo:ay	tamo:a
3.	tamo:ayona	tamo:ayo	tamo:ayo
4.	yamo:ay	yamo:ay	yamo:a
5.	tamo:ay	tamo:ay	tamo:a
6.	namo:ay	namo:ay	namo:a
7.	tamo:aya|ni	tamo:aya|	tamo:aya|
8.	yamo:aya|ni	yamo:aya|	yamo:aya|
8.	tamo:aya|ni	tamo:aya|	tamo:aya|
9.	tamo:awona	tamo:awo|	tamo:awo|
10.	tamo:ayona	tamo:ayona	tamo:ayona
11.	yamo:awona	yamo:awo|	yamo:awo|
12.	yamo:ayona	yamo:ayona	yamo:ayona

MPPAS

	indicative	**subjunctive**	**jussive**
1.	:umo:ay	:umo:ay	:umo:a
2.	tumo:ay	tumo:ay	tumo:a
3.	tumo:ayona	tumo:ayo	tumo:ayo
4.	yumo:ay	yumo:ay	yumo:a
5.	tumo:ay	tumo:ay	tumo:a
6.	numo:ay	numo:ay	numo:a
7.	tumo:aya|ni	tumo:aya|	tumo:aya|
8.	yumo:aya|ni	yumo:aya|	yumo:aya|
8.	tumo:aya|ni	tumo:aya|	tumo:aya|
9.	tumo:awona	tumo:awo|	tumo:awo|
10.	tumo:ayona	tumo:ayona	tumo:ayona
11.	yumo:awona	yumo:awo|	yumo:awo|
12.	yumo:ayona	yumo:ayona	yumo:ayona

(9) Weak; Faɛila-yafoɛalu: {gn, svq}; e.g., 'gn', "to become rich ...".

	PRACT	PRPAS
1.	ganiyotu	N/A.
2.	ganiyota	
3.	ganiyoti	
4.	ganiya	
5.	ganiyato	
6.	ganiyona|	
7.	ganiyotuma|	
8.	ganiya|	
8.	ganiyata|	
9.	ganiyotumo	
10.	ganiyotunna	
11.	ganuw|	
12.	ganiyona	

MPACT

	indicative	**subjunctive**	**jussive**
1.	:agonay	:agonay	:agona
2.	tagonay	tagonay	tagona
3.	tagonayona	tagonayo	tagonayo
4.	yagonay	yagonay	yagona
5.	tagonay	tagonay	tagona
6.	nagonay	nagonay	nagona
7.	tagonaya|ni	tagonaya|	tagonaya|
8.	yagonaya|ni	yagonaya|	yagonaya|
8.	tagonaya|ni	tagonaya|	tagonaya|
9.	tagonawona	tagonawo|	tagonawo|
10.	tagonayona	tagonayona	tagonayona
11.	yagonawona	yagonawo|	yagonawo|
12.	yagonayona	yagonayona	yagonayona

MPPAS
N/A.

(10) Weak; Faɛala-yafoɛulu: {:b, :x, ns}; e.g., ':x', "to befriend ...".

	PRACT	PRPAS
1.	:axawotu	:uxiyotu
2.	:axawota	:uxiyota
3.	:axawoti	:uxiyoti
4.	:axa|	:uxiya
5.	:axato	:uxiyato
6.	:axawona|	:uxiyona|
7.	:axawotuma|	:uxiyotuma|
8.	:axawa|	:uxiya|
8.	:axata|	:uxiyata|
9.	:axawotumo	:uxiyotumo
10.	:axawotunna	:uxiyotunna
11.	:axawo|	:uxuw|
12.	:axawona	:uxiyona

MPACT

	indicative	**subjunctive**	**jussive**
1.	:a:oxuw	:a:oxuwa	:a:oxu
2.	ta:oxuw	ta:oxuwa	ta:oxu
3.	ta:oxiyna	ta:oxiy	ta:oxiy
4.	ya:oxuw	ya:oxuwa	ya:oxu
5.	ta:oxuw	ta:oxuwa	ta:oxu
6.	na:oxuw	na:oxuwa	na:oxu
7.	ta:oxuwa|ni	ta:oxuwa|	ta:oxuwa|
8.	ya:oxuwa|ni	ya:oxuwa|	ya:oxuwa|
8.	ta:oxuwa|ni	ta:oxuwa|	ta:oxuwa|
9.	ta:oxuwna	ta:oxuw|	ta:oxuw|
10.	ta:oxuwona	ta:oxuwona	ta:oxuwona
11.	ya:oxuwna	ya:oxuw|	ya:oxuw|
12.	ya:oxuwona	ya:oxuwona	ya:oxuwona

MPPAS

	indicative	**subjunctive**	**jussive**
1.	:u:oxay	:u:oxay	:u:oxa
2.	tu:oxay	tu:oxay	tu:oxa
3.	tu:oxayona	tu:oxayo	tu:oxayo
4.	yu:oxay	yu:oxay	yu:oxa
5.	tu:oxay	tu:oxay	tu:oxa
6.	nu:oxay	nu:oxay	nu:oxa
7.	tu:oxaya|ni	tu:oxaya|	tu:oxaya|
8.	yu:oxaya|ni	yu:oxaya|	yu:oxaya|
8.	tu:oxaya|ni	tu:oxaya|	tu:oxaya|
9.	tu:oxawona	tu:oxawo|	tu:oxawo|
10.	tu:oxayona	tu:oxayona	tu:oxayona
11.	yu:oxawona	yu:oxawo|	yu:oxawo|
12.	yu:oxayona	yu:oxayona	yu:oxayona

d. Augmented Triliteral Defective Verbs

(1) Faɛɛala-yufaɛɛilu; Quasi-Sound: {h-h-d}

	PRACT	PRPAS
1.	wah-h-adotu	wuh-h-idotu
2.	wah-h-adota	wuh-h-idota
3.	wah-h-adoti	wuh-h-idoti
4.	wah-h-ada	wuh-h-ida
5.	wah-h-adato	wuh-h-idato
6.	wah-h-adona|	wuh-h-idona|
7.	wah-h-adotuma|	wuh-h-idotuma|
8.	wah-h-ada|	wuh-h-ida|
8.	wah-h-adata|	wuh-h-idata|
9.	wah-h-adotumo	wuh-h-idotumo
10.	wah-h-adotunna	wuh-h-idotunna
11.	wah-h-aduw|	wuh-h-iduw|
12.	wah-h-adona	wuh-h-idona

MPACT

	indicative	**subjunctive**	**jussive**
1.	:uwah-h-idu	:uwah-h-ida	:uwah-h-ido
2.	tuwah-h-idu	tuwah-h-ida	tuwah-h-ido
3.	tuwah-h-idiyna	tuwah-h-idiy	tuwah-h-idiy
4.	yuwah-h-idu	yuwah-h-ida	yuwah-h-ido
5.	tuwah-h-idu	tuwah-h-ida	tuwah-h-ido
6.	nuwah-h-idu	nuwah-h-ida	nuwah-h-ido
7.	tuwah-h-ida|ni	tuwah-h-ida|	tuwah-h-ida|
8.	yuwah-h-ida|ni	yuwah-h-ida|	yuwah-h-ida|
8.	tuwah-h-ida|ni	tuwah-h-ida|	tuwah-h-ida|
9.	tuwah-h-iduwna	tuwah-h-iduw|	tuwah-h-iduw|
10.	tuwah-h-idona	tuwah-h-idona	tuwah-h-idona
11.	yuwah-h-iduwna	yuwah-h-iduw|	yuwah-h-iduw|
12.	yuwah-h-idona	yuwah-h-idona	yuwah-h-idona

MPPAS

	indicative	**subjunctive**	**jussive**
1.	:uwah-h-adu	:uwah-h-ada	:uwah-h-ado
2.	tuwah-h-adu	tuwah-h-ada	tuwah-h-ado
3.	tuwah-h-adiyna	tuwah-h-adiy	tuwah-h-adiy
4.	yuwah-h-adu	yuwah-h-ada	yuwah-h-ado
5.	tuwah-h-adu	tuwah-h-ada	tuwah-h-ado
6.	nuwah-h-adu	nuwah-h-ada	nuwah-h-ado
7.	tuwah-h-ada|ni	tuwah-h-ada|	tuwah-h-ada|
8.	yuwah-h-ada|ni	yuwah-h-ada|	yuwah-h-ada|
8.	tuwah-h-ada|ni	tuwah-h-ada|	tuwah-h-ada|
9.	tuwah-h-aduwna	tuwah-h-aduw|	tuwah-h-aduw|
10.	tuwah-h-adona	tuwah-h-adona	tuwah-h-adona
11.	yuwah-h-aduwna	yuwah-h-aduw|	yuwah-h-aduw|
12.	yuwah-h-adona	yuwah-h-adona	yuwah-h-adona

(2) Faɛɛala-yufaɛɛilu; Hollow, e.g., i. {byyt, byyd;, s;yyr}; e.g., 'byyt', "to ponder overnight".

	PRACT	PRPAS
1.	bayyattu	buyyittu
2.	bayyatta	buyyitta
3.	bayyatti	buyyitti
4.	bayyata	buyyita
5.	bayyatato	buyyitato
6.	bayyatona|	buyyitona|
7.	bayyattuma|	buyyittuma|
8.	bayyata|	buyyita|
8.	bayyatata|	buyyitata|
9.	bayyattumo	buyyittumo
10.	bayyattunna	buyyittunna
11.	bayyatuw|	buyyituw|
12.	bayyatona	buyyitona

MPACT

	indicative	**subjunctive**	**jussive**
1.	:ubayyitu	:ubayyita	:ubayyito
2.	tubayyitu	tubayyita	tubayyito
3.	tubayyitiyna	tubayyitiy	tubayyitiy
4.	yubayyitu	yubayyita	yubayyito
5.	tubayyitu	tubayyita	tubayyito
6.	nubayyitu	nubayyita	nubayyito
7.	tubayyita|ni	tubayyita|	tubayyita|
8.	yubayyita|ni	yubayyita|	yubayyita|
8.	tubayyita|ni	tubayyita|	tubayyita|
9.	tubayyituwna	tubayyituw|	tubayyituw|
10.	tubayyitona	tubayyitona	tubayyitona
11.	yubayyituwna	yubayyituw|	yubayyituw|
12.	yubayyitona	yubayyitona	yubayyitona

MPPAS

	indicative	**subjunctive**	**jussive**
1.	:ubayyatu	:ubayyata	:ubayyato
2.	tubayyatu	tubayyata	tubayyato
3.	tubayyatiyna	tubayyatiy	tubayyatiy
4.	yubayyatu	yubayyata	yubayyato
5.	tubayyatu	tubayyata	tubayyato
6.	nubayyatu	nubayyata	nubayyato
7.	tubayyata|ni	tubayyata|	tubayyata|
8.	yubayyata|ni	yubayyata|	yubayyata|
8.	tubayyata|ni	tubayyata|	tubayyata|
9.	tubayyatuwna	tubayyatuw|	tubayyatuw|
10.	tubayyatona	tubayyatona	tubayyatona
11.	yubayyatuwna	yubayyatuw|	yubayyatuw|
12.	yubayyatona	yubayyatona	yubayyatona

(3) Faɛɛala-yufaɛɛilu; Hollow, e.g., ii. {:wwl, sww:, kwwn, qwwm, t;wwl, bwwb, xwwl, swwd, jwwd}; e.g., 'qwwm', "to make upright".

	PRACT	PRPAS
1.	qawwamotu	quwwimotu
2.	qawwamota	quwwimota
3.	qawwamoti	quwwimoti
4.	qawwama	quwwima
5.	qawwamato	quwwimato
6.	qawwamona|	quwwimona|
7.	qawwamotuma|	quwwimotuma|
8.	qawwama|	quwwima|
8.	qawwamata|	quwwimata|
9.	qawwamotumo	quwwimotumo
10.	qawwamotunna	quwwimotunna
11.	qawwamuw|	quwwimuw|
12.	qawwamona	quwwimona

MPACT

	indicative	**subjunctive**	**jussive**
1.	:uqawwimu	:uqawwima	:uqawwimo
2.	tuqawwimu	tuqawwima	tuqawwimo
3.	tuqawwimiyna	tuqawwimiy	tuqawwimiy
4.	yuqawwimu	yuqawwima	yuqawwimo
5.	tuqawwimu	tuqawwima	tuqawwimo
6.	nuqawwimu	nuqawwima	nuqawwimo
7.	tuqawwima|ni	tuqawwima|	tuqawwima|
8.	yuqawwima|ni	yuqawwima|	yuqawwima|
8.	tuqawwima|ni	tuqawwima|	tuqawwima|
9.	tuqawwimuwna	tuqawwimuw|	tuqawwimuw|
10.	tuqawwimona	tuqawwimona	tuqawwimona
11.	yuqawwimuwna	yuqawwimuw|	yuqawwimuw|
12.	yuqawwimona	yuqawwimona	yuqawwimona

MPPAS

	indicative	**subjunctive**	**jussive**
1.	:uqawwamu	:uqawwama	:uqawwamo
2.	tuqawwamu	tuqawwama	tuqawwamo
3.	tuqawwamiyna	tuqawwamiy	tuqawwamiy
4.	yuqawwamu	yuqawwama	yuqawwamo
5.	tuqawwamu	tuqawwama	tuqawwamo
6.	nuqawwamu	nuqawwama	nuqawwamo
7.	tuqawwama|ni	tuqawwama|	tuqawwama|
8.	yuqawwama|ni	yuqawwama|	yuqawwama|
8.	tuqawwama|ni	tuqawwama|	tuqawwama|
9.	tuqawwamuwna	tuqawwamuw|	tuqawwamuw|
10.	tuqawwamona	tuqawwamona	tuqawwamona
11.	yuqawwamuwna	yuqawwamuw|	yuqawwamuw|
12.	yuqawwamona	yuqawwamona	yuqawwamona

(4) Faɛɛala-yufaɛɛilu; Weak: {gnn, t-nn, bnn, d-kk}; e.g., 'gnn', "to sing ...".

	PRACT	PRPAS
1.	gannayotu	gunniyotu
2.	gannayota	gunniyota
3.	gannayoti	gunniyoti
4.	gannay	gunniya
5.	gannato	gunniyato
6.	gannayona|	gunniyona|
7.	gannayotuma|	gunniyotuma|
8.	gannaya|	gunniya|
8.	gannata|	gunniyata|
9.	gannayotumo	gunniyotumo
10.	gannayotunna	gunniyotunna
11.	gannuw|	gunnuw|
12.	gannayona	gunniyona

MPACT

	indicative	**subjunctive**	**jussive**
1.	:uganniy	:uganniya	:uganni
2.	tuganniy	tuganniya	tuganni
3.	tuganniyna	tuganniy	tuganniy
4.	yuganniy	yuganniya	yuganni
5.	tuganniy	tuganniya	tuganni
6.	nuganniy	nuganniya	nuganni
7.	tuganniya|ni	tuganniya|	tuganniya|
8.	yuganniya|ni	yuganniya|	yuganniya|
8.	tuganniya|ni	tuganniya|	tuganniya|
9.	tugannuwna	tugannuw|	tugannuw|
10.	tuganniyona	tuganniyona	tuganniyona
11.	yugannuwna	yugannuw|	yugannuw|
12.	yuganniyona	yuganniyona	yuganniyona

MPPAS

	indicative	**subjunctive**	**jussive**
1.	:ugannay	:ugannay	:uganna
2.	tugannay	tugannay	tuganna
3.	tugannayona	tugannayo	tugannayo
4.	yugannay	yugannay	yuganna
5.	tugannay	tugannay	tuganna
6.	nugannay	nugannay	nuganna
7.	tugannaya|ni	tugannaya|	tugannaya|
8.	yugannaya|ni	yugannaya|	yugannaya|
8.	tugannaya|ni	tugannaya|	tugannaya|
9.	tugannawona	tugannawo|	tugannawo|
10.	tugannayona	tugannayona	tugannayona
11.	yugannawona	yugannawo|	yugannawo|
12.	yugannayona	yugannayona	yugannayona

B. NOMINAL PATTERNING

a. LIST 1: L1 Pattern-Root Distribution for NUMBER/GENDER Categories

	BASE PATTERN	BP PATTERN	ROOT

a.i. Pattern-Root Distribution for the MF-GENDER Group

			Basic Triliteral Sound.
			Solid/'Mahomuwz':
1.	ca|cic	N/A.	{h-sn, krm, kbr, kt-r, s;gr, jml, drs, frsv, s;dq, jbl, h-mr, zrq, mnh-, xd;r, svbk, sds, tsɛ, s;fr, h-dq, fhm, mr:}.
2.	ca|:ic	ca|cat	Hollow: {sd}.
3.	ca|:ic	N/A.	Hollow: {kn, s;r, bt, bd;, t;l, jd, xl}.
4.	ca|wic	N/A.	Hollow: {:l}.
5.	ca|cc	cawa|cc	Doubled: {ɛmm}.
6.	ca|cc	N/A.	Doubled: {d/nn, jdd, :mm}.
7.	wa|cic	N/A.	Quasi-Sound: {ld, rd, sm}.
			Augmented Triliteral Sound.
			Solid/'Mahomuwz':
8.	mucaccic	N/A.	{kttb, drrs, frrsv, qttl, skkn, ftth-, jlls, kssr, d;rrb, nzzl, fhhm, ɛllm, svrrb, h-ssn, krrm, kbbr, kt-t-r, s;ggr, qs;s;r, jmml, fqqr, jddd, t-mmn, svrrf, rbbɛ, h-mml, qllm, svbbk, ɛsvsvr, sdds, sbbɛ, t-llt-, xmms, :llf, s;ffr, rkkb, mrr:, rjjl, s;ddq, nffd-, h-ddq, jbbl, xd;d;r, h-mmr, :mmm, ɛmmm}.
			Augmented Triliteral Defective.
9.	muwaccic	N/A.	Quasi-Sound: {ssm, lld, rrd, h-h-d}.
10.	mucayyic	N/A.	Hollow: {s;yyr, byyt, byyd;}.
11.	mucawwic	N/A.	Hollow: {:wwl, kwwn, qwwm, t;wwl, bwwb, jwwd}.

a.ii. Pattern-Root Distribution for the MO-GENDER Group

			Basic Triliteral Defective.
12.	ca|:ic	cuyya|c cuyyac	Hollow: {qm}.
13.	ca|c	cuca|t	Weak: {bn}.
14.	ca|c	N/A.	Weak: {t-n, m:, gn, d-k, svq, ns},
			Quasi-Sound: {h-d}.
15.	muca|c	N/A.	Weak: {:x}.
			Basic Triliteral Sound.
			Solid/'Mahomuwz':
16.	ca|cic	cacoc	{svrb}.
17.	ca|cic	cucuwc	{jls}.
18.	ca|cic	cacacat	{qlm, h-ml, h-sb, mnɛ, sbɛ}.
19.	ca|cic	cucuc cucuwc	{svrf, qdm}.
20.	ca|cic	cucca|c cacacat	{ktb, qtl}.
21.	ca|cic	cucca|c cucoca|n	{rjl, rkb}.
22.	ca|cic	cucca|c	{skn, ɛlm, :lf}.
23.	ca|cic	cuccac	{svkr, ksr}.
24.	ca|cic	N/A.	{nzl, d;rb, d-hb, fth-, nfd-, qs;r, t-lt-, t-mn, xms, rbɛ, ɛsvr}.
			Augmented Triliteral Defective.
25.	mucacc	N/A.	Weak: {bnn, t-nn, d-kk, gnn}.

a.iii. Pattern-Root Distribution for the FO-GENDER Group

			Basic Triliteral Defective.
26.	muca|ciy	N/A.	Weak: {:x}.
27.	ca|ciy	N/A.	Quasi-Sound: {h-d},
			Weak: {d-k, m:}.
28.	ca|ciy	cawa|c	Weak: {bn, t-n, gn, svq, ns}.
			Basic Triliteral Defective.
29.	ca|:ic	cawa|:ic	Hollow: {kn, qm}.
			Basic Triliteral Sound.
			Solid/'Mahomuwz':
30.	ca|cic	cawa|cic	{svrf, fth-, h-ml, qdm, qs;r, svrb, skn, nzl, d;rb, rkb, mnɛ,
			d-hb, nfd-, t-lt-, t-mn, sbɛ, xms, rbɛ, ɛsvr, :lf, ksr}.
31.	ca|cic	N/A.	{rjl, ktb, qtl, qlm, jls, svkr, h-sb, ɛlm}.
			Basic Triliteral Defective.
32.	mucacciy	N/A.	Weak: {bnn, t-nn, d-kk, gnn}.

b. LIST 2: L2 Pattern-Root Distribution for NUMBER/GENDER Categories

BASE PATTERN	BP PATTERN	ROOT	

b.i. Pattern-Root Distribution for the MF-GENDER Group

			Basic Triliteral Defective.
1.	macuwc	N/A.	Hollow: {kn, qm, sd}.
2.	macociyy	N/A.	Weak: {bn, t-n, m:, ns}.
3.	mawocuwc	N/A.	Quasi-Sound: {ld, rd, h-d, sm}.
			Basic Triliteral Sound.
			Solid/'Mahomuwz':
4.	macocuwc	N/A.	{ktb, svkr, h-sb, drs, frsv, qtl, rbɛ, skn, d/nn, fth-, mnh-,
			mnɛ, d-hb, jls, ksr, d;rb, nzl, h-ml, qlm, svbk, ɛsvr, sds,
			sbɛ, t-lt-, t-mn, tsɛ, xms, :lf, h-dq, fhm, ɛlm, qdm, svrb,
			rkb, mr:, jbl}.
			Augmented Triliteral Sound.
			Solid/'Mahomuwz':
5.	mucaccac	N/A.	{kttb, drrs, frrsv, qttl, skkn, ftth-, jlls, kssr, d;rrb, nzzl,
			fhhm, ɛllm, svrrb, h-ssn, krrm, kbbr, kt-t-r, s;ggr, qs;s;r,
			jmml, fqqr, jddd, t-mmn, svrrf, rbbɛ, h-mml, qllm, svbbk,
			ɛsvsvr, sdds, sbbɛ, t-llt-, xmms, :llf, s;ffr, rkkb, mrr:, rjjl,
			s;ddq, nffd-, h-ddq, jbbl, xd;d;r, h-mmr, :mmm, ɛmmm}.
			Augmented Triliteral Defective.
6.	muwaccac	N/A.	Quasi-Sound: {ssm, lld, rrd, h-h-d}.
7.	mucayyac	N/A.	Hollow: {s;yyr, byyt, byyd;}.
8.	mucawwac	N/A.	Hollow: {:wwl, kwwn, qwwm, t;wwl, bwwb, jwwd, xwwl,
			sww:, swwd}.

b.ii. Pattern-Root Distribution for the MO-GENDER Group

			Augmented Triliteral Defective.
9.	mucaccay	N/A.	Weak: {bnn, t-nn, d-kk, gnn}.
10.	muca|cay	N/A.	Weak: {:x}.

b.iii. Pattern-Root Distribution for the FO-GENDER Group

			Augmented Triliteral Defective.
11.	muca|ca|t	muca|cay	Weak: {:x}.
12.	mucacca|t	mucaccay	Weak: {bnn, t-nn, d-kk, gnn}.

c. LIST 3: Substantive Pattern-Stem Distribution for NUMBER/GENDER Categories

BASE PATTERN	BP PATTERN	STEM

c.i. Pattern-Stem Distribution for the MF-GENDER Group

			Sound:
1.	cacoc	N/A.	{mr:}.
2.	cacac	:acoca|c cica|c	{d-hb}.
3.	ca|cic	cawa|cic	{sɛd}.
4.	cica|c	caca|:ic	{rkb}.
5.	cica|c	:aciccat cayiccat	{:mm}.
6.	caciyc	caca|:ic	{ktb, h-mr, h-ml, mnh-}.
7.	cica|:	:acociyat	Defective: {bn}.

c.ii. Pattern-Stem Distribution for the MO-GENDER Group

			Sound:
8.	caciyc	N/A.	{s;fr}.
9.	caciyc	cucuc cica|c	{rbɛ}.
10.	caciyc	cucaca|: cucca|c	{jls}.
11.	caciyc	:acoca|c cicoca|n	{drs}.
12.	caciyc	:acocica|: :acocicat	{xms}.
13.	caciyc	:acicca|:	{d/nn}.
14.	cacuc	:aca|cic cica|c	{rjl}.
15.	micoca|c	maca|ciyc	{fth-}.
16.	cacac	:acoca|c :acocicat	{t-mn}.
17.	cica|c	cucoca|n	{h-sb}.
18.	cacac	:acoca|c	{h-sb}.
19.	cica|c	cucuc :acocicat	{h-mr}.
20.	caca|c	:acocicat	{svrb}.
21.	cucca|c	caca|ciyc	{svbk}.
22.	cacuc	cucuwc cica|c	{sbɛ}.
23.	cacac	:acoca|c cica|c	{jbl}.
24.	caciyc	cucuc	{jbl, svkr, tsɛ, rkb}.
25.	caciyc	:acoca|c	{t-lt-, sds, sbɛ}.
26.	caciyc	cucaca|:	{:lf, nzl, d;rb}.
27.	caciyc	:acocica|: cucaca|:	{s;dq, ɛsvr}.
28.	cica|c	cucuc	{ktb, frsv}.
29.	cacoc	cucuwc	{drs, qs;r}.
30.	cacac	:acoca|c cica|c	{qlm, ɛlm, jml}.
31.	cacc	:acoca|c cucuwcat	{ɛmm, jdd}.
			Defective:
32.	cayyic	:acoya|c ca|cat	{sd}.
33.	waciyc	:awocicat	{rd}.
34.	cac	ca|ca|:	{:b}.
35.	cac	ca|ca|: cicowat	{:x}.
36.	wica|c	:awocicat	{sm}.
37.	cayoc	:acoya|c cuyuwc	{bt}.

38.	wacac	:awoca|c wicocat	{ld}.
39.	waciyc	wicoca|n	{ld}.
40.	:icoc	:acoca|:	{bn}.
41.	cawa|c	:acoya|c ciya|c	{jd}.
42.	ca|c	:acowa|c :acowicat	{xl, bb}.
43.	cawoc	:acowa|c	{kn, qm}.

c.iii. Pattern-Stem Distribution for the FO-GENDER Group

			Sound:
44.	:icocac	N/A.	{mr:}.
45.	cucc	cucac	{:mm}.
46.	cuccah	N/A.	{:mm}.
47.	cacuc	N/A.	{sbɛ}.
48.	cacac	:acoca|c cica|c	{ɛlm}.
49.	cucoc	cucac	{svrf}.
50.	cicoc	cicac	{mnh-}.
51.	cacac	:acoca|c cuca|c	{qdm}.
52.	cacac	cica|c	{svbk}.
53.	cica|c	N/A.	{h-mr}.
54.	cica|c	caca|:ic	{ɛmm}.
55.	caciyc	caca|:ic	{h-dq, jbl, rbɛ, ɛsvr, :lf, skn, d;rb, frsv}.
56.	caciyc	N/A.	{jls, nzl, rkb, d/nn, s;dq}.
57.	cica|c	N/A.	{ktb, drs}.
58.	caca|c	N/A.	{s;dq, frsv}.
59.	cacac	:acoca|c cacac	{h-dq, qs;r}.
60.	cacc	N/A.	{ɛmm, jdd}.
			Defective:
61.	N/A.	cica|: cicowat	{ns}.
62.	cuc	N/A.	{:x}.
63.	cayyic	N/A.	{sd}.
64.	cic	N/A.	{bn}.
65.	:icoc	N/A.	{bn}.
66.	waciyc	waca|:ic	{ld}.
67.	wica|c	N/A.	{ld}.
68.	wacoc	wucuwc :awoca|c	{rd}.
69.	cayoc	cuyuwc cayoc	{bd;}.
70.	wicoc	N/A.	{h-d}.
71.	ca|c	ciyac	{qm}.
72.	ca|c	N/A.	{xl, bb}.

d. LIST 4: LOC Pattern-Stem Distribution for NUMBER/GENDER Categories

BASE PATTERN	BP PATTERN	STEM

d.i. Pattern-Stem Distribution for the MF-GENDER Group

			Sound:
1.	macocac	maca|cic	{ktb, drs, svrb, qs;r, rkb, rbɛ, sbɛ}.
2.	macocic	maca|cic	{nzl, d;rb}.
3.	micocac	maca|cic	{frsv}.
4.	maca|c	N/A.	Defective: {qm}.

d.ii. Pattern-Stem Distribution for the MO-GENDER Group

			Sound:
5.	macocac	maca|cic	{h-sn, svrf, d-hb, qdm, qtl, skn, jbl, fth-, ɛlm, ɛsvr}.
6.	macocic	maca|cic	{jls, ksr, h-ml, nfd-}.
7.	micocac	maca|cic	{drs, xd;r, qlm, rjl, h-mr, t-lt-, t-mn}.
			Defective:
8.	maca|c	:ama|cic :amocicat	{kn}.
9.	maciyc	maca|yic	{s;r}.
10.	maciyc	N/A.	{bt}.
11.	mawocic	mawa|cic	{ld, rd, sm}.
12.	macocay	maca|c	{bn, t-n, gn}.

d.iii. Pattern-Stem Distribution for the FO-GENDER Group

			Sound:
13.	macocic	maca|cic	{h-sb}.
14.	macocac	N/A.	{xd;r}.
15.	macicc	maca|cc	{d/nn}.
16.	micocac	maca|cic	{t-mn, svrb}.
17.	maca|c	N/A.	Defective: {kn, t-n}.

e. LIST 5: AA Pattern-Stem Distribution for NUMBER/GENDER Categories

BASE PATTERN	**BP PATTERN**	**STEM**

e.i. Pattern-Stem Distribution for the MF-GENDER Group

			Sound:
1.	caciyc	cica|c	{qs;r}.
2.	caciyc	N/A.	{kt-r}.
3.	cacac	cica|c	{h-sn}.
4.	caciyc	caca|:ic	{:mm, qdm, mnɛ, d-hb}.
			Defective:
5.	waciyc	N/A.	{sm, h-d}.
6.	cayyic	N/A.	{s:}.
7.	cawiyc	ciya|c	{qm}.

e.ii. Pattern-Stem Distribution for the MO-GENDER Group

			Sound:
8.	caciyc	cucuwc :acocicat	{mr:}.
9.	caciyc	cucaca|: :acoca|c	{svrf}.
10.	caciyc	cucaca|: cica|c	{svrb, kbr, krm, s;gr}.
11.	caciyc	cucaca|:	{fqr, qs;r, sɛd, jml, h-sb, ɛlm}.
12.	caciyc	cucuc	{jdd, t-mn, nfd-}.
13.	caciyc	cacocay caca|cay	{ksr, rjl}.
14.	caciyc	cucaca|: cuca|cay	{qtl, qdm}.
15.	:acocac	cucoc	{xd;r, h-mr, zrq, s;fr}.
			Defective:
16.	cayyic	caya|:ic	{jd}.
17.	cawiyc	ciwa|c ciya|c	{t;l}.
18.	waciyc	wucaca|:	{sm}.

19. cawwac cawa|:ic {:l}.
20. :acowac cuwc {sd}.
21. :acoyac ciyc {bd;}.
22. caciyy :acociya|: {gn, svq, d-k}.

e.iii. Pattern-Stem Distribution for the FO-GENDER Group

			Sound:
23.	caciyc	N/A.	{svrb, qtl, ksr, sɛd, h-sb, ɛlm, nfd-, jdd, rjl, t-mn, mr:}.
24.	caciyc	caca|:ic	{krm, kbr, qs;r, s;gr, fqr, jml, svrf}.
25.	cacoca|:	N/A.	{xd;r, h-mr, zrq, s;fr}.
			Defective:
26.	caciyy	N/A.	{d-k, svq, gn}.
27.	cayyic	N/A.	{jd}.
28.	cuwcay	cuwac	{:l}.
29.	cawiyc	N/A.	{t;l}.
30.	cawoca|:	N/A.	{sd}.
31.	cayoca|:	N/A.	{bd;}.

f. LIST 6: MM Pattern-Stem Distribution for NUMBER/GENDER Categories

BASE PATTERN	BP PATTERN	STEM

f.i. Pattern-Stem Distribution for the MF-GENDER Group

			Sound:
1.	cacoc	N/A.	{sbɛ, xms}.
2.	caca|c	N/A.	{t-lt-}.
3.	:acocac	N/A.	{rbɛ}.
4.	cicoc	N/A.	{tsɛ}.
5.	cicc	N/A.	{stt}.
6.	wa|cic	N/A.	Defective: {h-d}.

f.ii. Pattern-Stem Distribution for the MO-GENDER Group

				Sound:
7.	caca|c	N/A.		{t-mn}.
8.	cacoc	N/A.		{ɛsvr}.
9.	cacoc	ca|ca|c	cucuwc	{:lf}.
10.	:acac	ca|ca|c		Defective: {h-d}.

f.iii. Pattern-Stem Distribution for the FO-GENDER Group

			Sound:
11.	caca|ciy	N/A.	{t-mn}.
12.	cacac	N/A.	{ɛsvr}.
			Defective:
13.	cic	N/A.	{m:}.
14.	:icocay	N/A.	{h-d}.

f.iv. Pattern-Stem Distribution for the Dual Group

15. :icoc N/A. Defective: {t-n}.

C. EXAMPLES OF MORPHOLOGICAL CATEGORIES

DESCRIPTORS				EXAMPLES
VB	CVB	QVB	QCVB	'yakotubu', "he writes" ...
PVB	CPVB	QPVB	QCPVB	'sayakotubu', "he is going to write" ...
VR	CVR	QVR	QCVR	'katabahu', "he wrote it" ...
PVR	CPVR	QPVR	QCPVR	'liyakotubahu', "in order to write it", 'sayad;oribuhu', "he is going to hit him" ...
VN	CVN	QVN	QCVN	'd-ahaba', "he went/(the) gold (of)" ...
VA	CVA			':axod;ara', "I became green/green" ...
VM	CVM			'ɛasvara', "he made ... tenfold/ten" ...
WV	CWV			'h-asabahu', "he counted it/his esteem" ...
NN	CNN	QNN	QCNN	'kita|bu+', "a book", 'makotabu+', "a desk" ...
DN	CDN	QDN	QCDN	'muwsay', 'zayodu+', "zayod", ':alrrajulu', "the man", ':alokita|bu', "the book", ':alomakotabu', "the desk", 'had-a|', "this", ':aloka|tibu', "the writer", ':alomakotuwbu', "the written" ...
PNN	CPNN	QPNN	QCPNN	'bikita|bi+', "in a book", 'bimakotabi+', "at a desk" ...
PDN	CPDN	QPDN	QCPDN	'bimuwsay', "with muwsay", 'bizayodi+', "with zayod", 'bi:alrrajuli', "with the man", 'bi:alokita|bi', "in the book", 'bi:alomakotabi', "at the desk", 'bihad-a|', "with this", 'bi:aloka|tibi', "with the writer", 'bi:alomakotuwbi', "in the written" ...
MM	CMM	QMM	QCMM	'saboɛu', "seven (of) ...", 'saboɛu+', "seven" ...
DM	CDM	QDM	QCDM	':aloxamosu', "the five" ...
PMM	CPMM	QPMM	QCPMM	'bixamosi+', "with five" ...
PDM	CPDM	QPDM	QCPDM	'bi:aloxamosi', "with the five" ...
AA	CAA	QAA	QCAA	'kabiyru+', "big, senior" ...
DA	CDA	QDA	QCDA	':alokabiyru', "the big/senior" ...
PAA	CPAA	QPAA	QCPAA	'bikabiyri+', "with a big/senior" ...
PDA	CPDA	QPDA	QCPDA	'bi:alokabiyri', "with the big/senior" ...
L1	CL1	QL1	QCL1	'ka|tibu+', "writing, a writer"...
PL1	CPL1	QPL1	QCPL1	'lika|tibi+', "for a writer"...
L2	CL2	QL2	QCL2	'makotuwbu+', "written"...
PL2	CPL2	QPL2	QCPL2	'limakotuwbi+', "for a written"...
MD	CMD	QMD	QCMD	'kita|bu', "(the) book (of)", 'ka|tibu', "(the) writer (of)", 'mamonuwh-u', "(the) recipient (of)", 'kabiyru', "big/senior" ...
PMD	CPMD	QPMD	QCPMD	'bikita|bi', "in (the) book (of)", 'bika|tibi', "with (the) writer (of)", 'bimamonuwh-i', "with (the) recipient (of)", 'bikabiyri', "with (the) big/senior (of)" ...
AN	CAN	QAN	QCAN	'makotabuhu', "his desk", 'kita|buhu', "his book", 'ka|tibuhu', "its writer",

				'mamonuwh-uhu', "his recipient", 'kabiyruhu', "his big/senior" ...
PAN	CPAN	QPAN	QCPAN	'bimakotabihi', "at his desk", 'bikita|bihi', "in his book", 'bika|tibihi', "with its writer", 'bimamonuwh-ihi', "with his recipient", 'bikabiyrihi', "with his big/senior" ...
XN	CXN	QXN	QCXN	'mabonay', "(the) building (of)/a building" ...
PXN	CPXN	QPXN	QCPXN	'limabonay', "to (the) building (of)/to a building" ...
XA	CXA			':uwlay', "(the) first (of)/first", ':ih-oday', "one (of)/one" ...
WN	CWN	QWN	QCWN	'ka|tibiy', "(the) writers (of)/my writer" ...
PWN	CPWN	QPWN	QCPWN	'bika|tibiy', "with (the) writers (of)/with writer"...
JP	CJP	QJP	QCJP	'lamo', "not (with past reference)".
FD	CFD	QFD	QCFD	'sawofa', "will, shall".
PD	CPD	QPD	QCPD	'qado', "perhaps, already".
SP	CSP	QSP	QCSP	'lano', "never (with future reference)".
LD	CLD	QLD	QCLD	'huna|', "here" ...
AD	CAD	QAD	QCAD	':alo:a|na', "now" ...
Q2	CQ2			'halo', "whether (with a question)".
JQ	CJQ	QJQ	QCJQ	'ma|', "what".
EP	CEP	QEP	QCEP	'la|', "no, not (with present reference)".
HQ	CHQ	QHQ	QCHQ	'mano', "who", 'limano', "to whom", 'kamano', "like whom" ...
QP	PQP	CQP		':ayona', "where" ...
Pr	CPr	QPr	QCPr	'fiy', "in, at"...
PGP	CPGP	QPGP	QCPGP	'laka', "to you" ...
AP	CAP	QAP	QCAP	':inna', "indeed, it is true that" ...

==0==<*****>==0==

Appendix B
THE DATABASES

1. Program Specifications and Statistics

PROGRAMMING LANGUAGE:
- ◇ LISP: Cambridge Orion 1.05/UNIX LISP System, entered in about 1550 Kbytes (FITCH and NORMAN).
- ◇ Store Image: made at 21:08:20, on 12 Apr 1990.
- ◇ LISP Version: 1.10/1.10/639950892.
- ◇ Image Size: 92,340 bytes.
- ◇ RAM required: normally run in about 1 megabyte.

PROCESSOR:
- ◇ CLIPPER: 32-Bit Microprocessor, made by INTERGRAPH.
- ◇ IMPLEMENTATION: on 3 CMOS VLSI chips.
- ◇ EXECUTION: one 30 nanosecond clock cycle (1 nanosecond = 1,000 millionth of a second) for most frequently used instructions.

OPERATING SYSTEM: Unix, Berkeley 4.2.

MACHINE:
- ◇ NAME: Xenakis.
- ◇ SYSTEM: HLH Orion.
- ◇ RAM: 8 megabytes.
- ◇ HARD DISC: 600 megabytes.

TUNIS1:
- ◇ DATABASE: 44,775 bytes.
- ◇ COMPILED: in 18,260 milliseconds.
- ◇ MAIN PROGRAM: 195,617 bytes.
- ◇ COMPILED: in 71,920 milliseconds.
- ◇ TOTAL SIZE: 240,392 bytes.

2. The Lexicon

LENGTH: 522.
value: (:alo :al lo l fiy mino :ilay :ilayo ɛano ɛalay ɛalayo maɛa ɛinoda laday ladayo fawoqa baɛoda tah-ota :ama|ma qabola wara|:a li la l bi b ka wa fa la| ma| lamo lano h-attay kayo :inna :anna sa sawofa qado :a halo matay :ayona kayofa ma|d-a| ma| mano huna| huna|ka huna|lika :alo:a|na :abada+ baɛodu at a|t a|ni a| ayoni ayo uwna uw iyna iy : n t y u

a o uwna uw| iyna iy a|ni a| ona na otu tu ona| na| ota ta oti ti otuma| tuma| otumo tumo otunna tunna ona na a ato a| ata| uw| s;a|lih- zayod ɛamor hisva|m mlayek ɛiysay muwsay yah-oyay riym lat;iyfat zakiyyat ɛawa|t;if zuhorat d;uh-ay layolay :ana| nah-onu :anota :anoti :anotuma| :anotumo :anotunna huwa hiya huma| humo hunna niy iy ya na| ka ki kuma| kumo kunna hu hi a| huma| hima| humo himo hunna hinna had-a| had-ihi had-a|ni had-ayoni ha|ta|ni ha|tayoni ha:ula|:i ls kn s;r bt bd; qm t;l bb jd xl s: sd bn t-n d-k m: gn svq :b :x ns ld rd h-d sm :l d/nn jdd ɛmm h-sb h-sn krm kbr kt-r s;gr qs;r jml fqr t-mn svrf ktb svkr drs frsv qtl rbɛ skn fth- mnh- mnɛ d-hb xd;r jls ksr d;rb nzl h-ml qlm svbk ɛsvr sds stt sbɛ t-lt- tsɛ xms :lf s;fr h-dq fhm ɛlm qdm svrb rkb mr: sɛd rjl s;dq nfd-jbl h-mr zrq :mm kttb drrs frrsv qttl skkn ftth- jlls kssr d;rrb nzzl fhhm ɛllm svrrb h-ssn krrm kbbr kt-t-r s;ggr qs;s;r jmml fqqr jddd t-mmn svrrf rbbɛ h-mml qllm svbbk ɛsvsvr sdds sbbɛ t-llt- xmms :llf s;ffr rkkb mrr: rjjl s;ddq nffd- h-ddq jbbl xd;d;r h-mmr :mmm ɛmmm bnn t-nn d-kk gnn ssm lld rrd h-h-d s;yyr byyt byyd; :wwl kwwn qwwm t;wwl bwwb jwwd xwwl sww: swwd mucacca|t muca|ca|t muca|cay macocuwc mawocuwc macuwc mucaccay ucaccac mucaccic macociyy mucacciy mucacc muca|ciy muca|c mucayyac mucayyic muwaccac mucawwac mucawwic muwaccic uwaccac uwaccic ucawwic ucawwac ucayyac ucayyic ucaccic ucacciy ucaccac ucocac ucaccay ucocay ucacc uwcac ucoc acocuc acocac acocic acociy acocay acacc acucc aciyc acoc acic acuc acocuw awocuc acowac acuwc cuccic cuyyic cayyac cucciy cuciy cucic caccac cacic caciy cacaw caccay cacay cuwwic cawic wacuc wuccic waccac wucic ca|:ic ca|wic ca|ciy ca|cc ciyc cic cicc cac cacc cuc cucc cuccah cucuc cucoc cucac cuyyac cuyya|c cuccac cuca|t wucuwc cuyuwc cuwcay cuwc cuwac cucuwcat cawoc cawwac cayoc cacoc :acac cacac cacuc cicoc cicac ciyac :icoc :icocay :icocac ca|c wa|cic ca|cic ca|cat cacacat ca|ca|: ca|ca|c caca|c cacocay caca|cay cuca|cay cayiccat :aciccat waca|:ic cawa|c cawa|cc caya|:ic cawa|:ic cawa|cic caca|:ic :awoca|c :acowa|c :acoya|c :acoca|: :acoca|c :acowac :acoyac :acocac :acocicat cawoca|: cayoca|: cacoca|: :acociya|: :acicca|: :acocica|: :aca|cic wicoc wacoc wacac waciyc cawiyc cayyic caciyy caciyc wica|c ciya|c ciwa|c cica|: cica|c wicocat cicowat :acociyat :awocicat :acowicat wucaca|: cuca|c cucca|c cucaca|: wicoca|n cicoca|n cucoca|n micoca|c caca|ciyc maca|ciyc maca|cic macicc mawocic macocic macocay macocac micocac maciyc maca|cc maca|c maca|yic mawa|cic :ama|cic :amocicat nil o a u i a+ u+ i+ w y l : b t t- j h- x d- r z s sv s; d; t; d/ ɛ g f q k l m n h)

3. The Rootlist

LENGTH: 149.
value: (ls kn s;r bt bd; qm t;l bb jd xl s: sd bn t-n d-k m: gn svq :b :x ns ld rd h-d sm :l d/nn jdd ɛmm h-sb h-sn krm kbr kt-r s;gr qs;r jml fqr t-mn svrf ktb svkr drs frsv qtl rbɛ skn fth-mnh- mnɛ d-hb xd;r jls ksr d;rb nzl h-ml qlm svbk ɛsvr sds stt sbɛ t-lt- tsɛ xms :lf s;fr h-dq fhm ɛlm qdm svrb rkb mr: sɛd rjl s;dq nfd- jbl h-mr zrq :mm kttb drrs frrsv qttl skkn ftth- jlls kssr d;rrb nzzl fhhm ɛllm svrrb h-ssn krrm kbbr kt-t-r s;ggr qs;s;r jmml fqqr jddd t-mmn svrrf rbbɛ h-mml qllm svbbk ɛsvsvr sdds sbbɛ t-llt- xmms :llf s;ffr rkkb mrr: rjjl s;ddq nffd- h-ddq jbbl xd;d;r h-mmr :mmm ɛmmm bnn t-nn d-kk gnn ssm lld rrd h-h-d s;yyr byyt byyd; :wwl kwwn qwwm t;wwl bwwb jwwd xwwl sww: swwd)

4. The Pattern List

LENGTH: 185.
value: (mucacca|t muca|ca|t muca|cay macocuwc mawocuwc macuwc mucaccay ucaccac mucaccic macociyy mucacciy mucacc muca|ciy muca|c mucayyac mucayyic muwaccac mucawwac mucawwic muwaccic uwaccac uwaccic ucawwic ucawwac ucayyac ucayyic ucaccic ucacciy ucaccac ucocac ucaccay ucocay ucacc uwcac ucoc acocuc acocac acocic acociy acocay acacc acucc aciyc acoc acic acuc acocuw awocuc acowac acuwc cuccic cuyyic cayyac cucciy cuciy cucic caccac cacic caciy cacaw caccay cacay cuwwic cawic wacuc wuccic waccac wucic ca|:ic ca|wic ca|ciy ca|cc ciyc cic cicc cac cacc cuc cucc cuccah cucuc cucoc cucac cuyyac cuyya|c cuccac cuca|t wucuwc cuyuwc cuwcay cuwc cuwac cucuwcat cawoc cawwac cayoc cacoc :acac

cacac cacuc cicoc cicac ciyac :icoc :icocay :icocac ca|c wa|cic ca|cic ca|cat cacacat ca|ca|: ca|ca|c caca|c cacocay caca|cay cuca|cay cayiccat :aciccat waca|:ic cawa|c cawa|cc caya|:ic cawa|:ic cawa|cic caca|:ic :awoca|c :acowa|c :acoya|c :acoca|: :acoca|c :acowac :acoyac :acocac :acocicat cawoca|: cayoca|: cacoca|: :acociya|: :acicca|: :acocica|: :aca|cic wicoc wacoc wacac waciyc cawiyc cayyic caciyy caciyc wica|c ciya|c ciwa|c cica|: cica|c wicocat cicowat :acociyat :awocicat :acowicat wucaca|: cuca|c cucca|c cucaca|: wicoca|n cicoca|n cucoca|n micoca|c caca|ciyc maca|ciyc maca|cic macicc mawocic macocic macocay macocac micocac maciyc maca|cc maca|c maca|yic mawa|cic :ama|cic :amocicat nil)

5. Table of Errors Detectable by TUNIS1

"error1: Verb-spelling violation: c_1c_2 or ec! in:"
"error2: Verb-spelling violation: c_1oc_1! in:"
"error3: incompatible vowel-sequence: i/y + v $\neq$ i! in:"
"error4: incompatible vowel-sequence: u/a/w/| + v $\neq$ u! in:"
"error5: vowel-sequence violation: v1v2 or ev! in:"
"error6: illegal morph-combination: VB-CP; TRANSITIVITY violation in:"
"error7: illegal morph-combination: VL-CP; TRANSITIVITY violation in:"
"error8: incompatible vowel-sequence: yo + v $\neq$ i! in:"
"error9: incompatible vowel-sequence: o + v $\neq$ u! in:"
"!*error: "note : unknown word!:"

A. THE V-DATABASE

a. A Dictionary of V-Roots

	NBR	GDR	PRS	CAT	MDG	VOC	TRVY	MMK	REF
ls	Ø	Ø	Ø	VB	Ø	Ø	ctr	Ø	Ø
kn	Ø	Ø	Ø	VB	Ø	Ø	ctr	Ø	Ø
s;r	Ø	Ø	Ø	VB	Ø	Ø	ctr	Ø	Ø
bt	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
bd;	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
qm	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
t;l	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
bb	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
jd	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
xl	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
s:	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
sd	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
bn	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
t-n	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
d-k	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
m:	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
gn	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
svq	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
:b	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
:x	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
ns	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
ld	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
rd	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
h-d	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
sm	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
:l	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
d/nn	Ø	Ø	Ø	VB	Ø	Ø	xtr	Ø	Ø
jdd	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
ɛmm	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø

h-sb	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
h-sn	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
krm	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
kbr	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
kt-r	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
s;gr	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
qs;r	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
jml	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
fqr	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
t-mn	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
svrf	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
ktb	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
svkr	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
drs	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
frsv	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
qtl	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
rbɛ	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
skn	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
fth-	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
mnh-	Ø	Ø	Ø	VB	Ø	Ø	dtr	Ø	Ø
mnɛ	Ø	Ø	Ø	VB	Ø	Ø	ptr2	Ø	Ø
d-hb	Ø	Ø	Ø	VB	Ø	Ø	ptr1	Ø	Ø
xd;r	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
jls	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
ksr	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
d;rb	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
nzl	Ø	Ø	Ø	VB	Ø	Ø	ptr1	Ø	Ø
h-ml	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
qlm	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
svbk	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
ɛsvr	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
sds	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
stt	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
sbɛ	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
t-lt-	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
tsɛ	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
xms	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
:lf	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
s;fr	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
h-dq	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
fhm	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
ɛlm	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
qdm	Ø	Ø	Ø	VB	Ø	Ø	ptr1	Ø	Ø
svrb	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
rkb	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
mr:	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
sɛd	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
rjl	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
s;dq	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
nfd-	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
jbl	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
h-mr	Ø	Ø	Ø	VB	Ø	Ø	mtr	Ø	Ø
zrq	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
:mm	Ø	Ø	Ø	VB	Ø	Ø	ntr	Ø	Ø
kttb	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
drrs	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø

frrsv	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
qttl	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
skkn	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
ftth-	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
jlls	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
kssr	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
d;rrb	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
nzzl	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
fhhm	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
ϵllm	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
svrrb	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
h-ssn	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
krrm	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
kbbr	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
kt-t-r	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
s;ggr	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
qs;s;r	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
jmml	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
fqqr	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
jddd	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
t-mmn	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
svrrf	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
rbbϵ	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
h-mml	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
qllm	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
svbbk	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
ϵsvsvr	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
sdds	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
sbbϵ	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
t-llt-	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
xmms	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
:llf	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
s;ffr	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
rkkb	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
mrr:	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
rjjl	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
s;ddq	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
nffd-	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
h-ddq	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
jbbl	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
xd;d;r	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
h-mmr	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
:mmm	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
ϵmmm	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
bnn	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
t-nn	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
d-kk	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
gnn	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
ssm	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
lld	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
rrd	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
h-h-d	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
s;yyr	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
byyt	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
byyd;	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
:wwl	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø

kwwn	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
qwwm	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
t;wwl	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
bwwb	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
jwwd	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
xwwl	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
sww:	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø
swwd	Ø	Ø	Ø	VB	Ø	Ø	Ø	Ø	Ø

b. A Dictionary of V-Patterns

	PRACTP	PRPASP	MPACTP	MPPASP
ls	(cac cayoc)	Ø	Ø	Ø
kn	(cuc ca|c)	Ø	(acuwc acuc)	Ø
s;r	(cic ca|c)	Ø	(aciyc acic)	Ø
bt	(cic ca|c)	Ø	(aciyc acic)	Ø
bd;	(cic ca|c)	Ø	(aciyc acic)	Ø
qm	(cuc ca|c)	Ø	(acuwc acuc)	Ø
t;l	(cuc ca|c)	Ø	(acuwc acuc)	Ø
bb	(cuc ca|c)	Ø	(acuwc acuc)	Ø
jd	(cuc ca|c)	Ø	(acuwc acuc)	Ø
xl	(cuc ca|c)	Ø	(acuwc acuc)	Ø
s:	(cuc ca|c)	Ø	(acuwc acuc)	Ø
sd	(cuc ca|c)	Ø	(acuwc acuc)	Ø
bn	(cacay cac)	(cuciy cuc)	(acociy acoc)	(ucocay ucoc)
t-n	(cacay cac)	(cuciy cuc)	(acociy acoc)	(ucocay ucoc)
d-k	(cacay cac)	Ø	(acocay acoc)	Ø
m:	(cacay cac)	(cuciy cuc)	(acocay acoc)	(ucocay ucoc)
gn	(caciy cac)	Ø	(acocay acoc)	Ø
svq	(caciy cac)	Ø	(acocay acoc)	Ø
:b	(cacaw cac)	Ø	(acocuw acoc)	Ø
:x	(cacaw cac)	(cuciy cuc)	(acocuw acoc)	(ucocay ucoc)
ns	(cacaw cac)	Ø	(acocuw acoc)	Ø
ld	(wacac)	(wucic)	(acic)	(uwcac)
rd	(wacac)	Ø	(acic)	Ø
h-d	(wacac)	Ø	(acic)	Ø
sm	(wacuc)	Ø	(awocuc)	Ø
:l	(cawic)	Ø	(acowac)	Ø
d/nn	(cacac cacc)	(cucic cucc)	(acucc acocuc)	(ucacc ucocac)
jdd	(cacac cacc)	Ø	(acucc acocuc)	Ø
ɛmm	(cacac cacc)	(cucic cucc)	(acucc acocuc)	(ucacc ucocac)
h-sb	(cacac)	(cucic)	(acocuc)	(ucocac)
h-sn	(cacuc)	Ø	(acocuc)	Ø
krm	(cacuc)	Ø	(acocuc)	Ø
kbr	(cacuc)	Ø	(acocuc)	Ø
kt-r	(cacuc)	Ø	(acocuc)	Ø
s;gr	(cacuc)	Ø	(acocuc)	Ø
qs;r	(cacuc)	Ø	(acocuc)	Ø
jml	(cacuc)	Ø	(acocuc)	Ø
fqr	(cacuc)	Ø	(acocuc)	Ø
t-mn	(cacuc)	(cucic)	(acocuc)	(ucocac)
svrf	(cacuc)	Ø	(acocuc)	Ø
ktb	(cacac)	(cucic)	(acocuc)	(ucocac)
svkr	(cacac)	(cucic)	(acocuc)	(ucocac)
drs	(cacac)	(cucic)	(acocuc)	(ucocac)
frsv	(cacac)	(cucic)	(acocuc)	(ucocac)
qtl	(cacac)	(cucic)	(acocuc)	(ucocac)

rbɛ	(cacac)	(cucic)	(acocuc)	(ucocac)
skn	(cacac)	(cucic)	(acocuc)	(ucocac)
fth-	(cacac)	(cucic)	(acocac)	(ucocac)
mnh-	(cacac)	(cucic)	(acocac)	(ucocac)
mnɛ	(cacac)	(cucic)	(acocac)	(ucocac)
d-hb	(cacac)	(cucic)	(acocac)	(ucocac)
xd;r	(cacac)	Ø	(acocac)	Ø
jls	(cacac)	(cucic)	(acocic)	(ucocac)
ksr	(cacac)	(cucic)	(acocic)	(ucocac)
d;rb	(cacac)	(cucic)	(acocic)	(ucocac)
nzl	(cacac)	(cucic)	(acocic)	(ucocac)
h-ml	(cacac)	(cucic)	(acocic)	(ucocac)
qlm	(cacac)	(cucic)	(acocic)	(ucocac)
svbk	(cacac)	(cucic)	(acocic)	(ucocac)
ɛsvr	(cacac)	(cucic)	(acocic)	(ucocac)
sds	(cacac)	(cucic)	(acocic)	(ucocac)
stt	Ø	Ø	Ø	Ø
sbɛ	(cacac)	(cucic)	(acocic)	(ucocac)
t-lt-	(cacac)	(cucic)	(acocic)	(ucocac)
tsɛ	(cacac)	(cucic)	(acocic)	(ucocac)
xms	(cacac)	(cucic)	(acocic)	(ucocac)
:lf	(cacac)	(cucic)	(acocic)	(ucocac)
s;fr	(cacac)	Ø	(acocic)	Ø
h-dq	(cacac)	(cucic)	(acocic)	(ucocac)
fhm	(cacic)	(cucic)	(acocac)	(ucocac)
ɛlm	(cacic)	(cucic)	(acocac)	(ucocac)
qdm	(cacic)	(cucic)	(acocac)	(ucocac)
svrb	(cacic)	(cucic)	(acocac)	(ucocac)
rkb	(cacic)	(cucic)	(acocac)	(ucocac)
mr:	(cacic)	Ø	(acocac)	Ø
sɛd	(cacic)	Ø	(acocac)	Ø
rjl	(cacic)	Ø	(acocac)	Ø
s;dq	(cacac)	(cucic)	(acocuc)	(ucocac)
nfd-	(cacac)	(cucic)	(acocuc)	(ucocac)
jbl	(cacac)	(cucic)	(acocuc)	(ucocac)
h-mr	(cacac)	(cucic)	(acocuc)	(ucocac)
zrq	(cacac)	Ø	(acocuc)	Ø
:mm	(cacac cacc)	Ø	(acacc acocuc)	Ø
kttb	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
drrs	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
frrsv	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
qttl	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
skkn	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
ftth-	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
jlls	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
kssr	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
d;rrb	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
nzzl	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
fhhm	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
ɛllm	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
svrrb	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
h-ssn	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
krrm	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
kbbr	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
kt-t-r	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
s;ggr	(caccac)	(cuccic)	(ucaccic)	(ucaccac)

qs;s;r	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
jmml	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
fqqr	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
jddd	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
t-mmn	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
svrrf	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
rbbɛ	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
h-mml	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
qllm	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
svbbk	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
ɛsvsvr	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
sdds	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
sbbɛ	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
t-llt-	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
xmms	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
:llf	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
s;ffr	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
rkkb	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
mrr:	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
rjjl	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
s;ddq	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
nffd-	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
h-ddq	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
jbbl	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
xd;d;r	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
h-mmr	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
:mmm	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
ɛmmm	(caccac)	(cuccic)	(ucaccic)	(ucaccac)
bnn	(caccay cacc)	(cucciy cucc)	(ucacciy ucacc)	(ucaccay ucacc)
t-nn	(caccay cacc)	(cucciy cucc)	(ucacciy ucacc)	(ucaccay ucacc)
d-kk	(caccay cacc)	Ø	(ucacciy ucacc)	Ø
gnn	(caccay cacc)	(cucciy cucc)	(ucacciy ucacc)	(ucaccay ucacc)
ssm	(waccac)	Ø	(uwaccic)	Ø
lld	(waccac)	(wuccic)	(uwaccic)	(uwaccac)
rrd	(waccac)	Ø	(uwaccic)	Ø
h-h-d	(waccac)	(wuccic)	(uwaccic)	(uwaccac)
s;yyr	(cayyac)	(cuyyic)	(ucayyic)	(ucayyac)
byyt	(cayyac)	(cuyyic)	(ucayyic)	(ucayyac)
byyd;	(cayyac)	(cuyyic)	(ucayyic)	(ucayyac)
:wwl	(cawwac)	(cuwwic)	(ucawwic)	(ucawwac)
kwwn	(cawwac)	(cuwwic)	(ucawwic)	(ucawwac)
qwwm	(cawwac)	(cuwwic)	(ucawwic)	(ucawwac)
t;wwl	(cawwac)	(cuwwic)	(ucawwic)	(ucawwac)
bwwb	(cawwac)	(cuwwic)	(ucawwic)	(ucawwac)
jwwd	(cawwac)	(cuwwic)	(ucawwic)	(ucawwac)
xwwl	(cawwac)	(cuwwic)	(ucawwic)	(ucawwac)
sww:	(cawwac)	(cuwwic)	(ucawwic)	(ucawwac)
swwd	(cawwac)	(cuwwic)	(ucawwic)	(ucawwac)

%%% in addition the roots 'h-sb', and 't-mn' have these patterns:

	PACP	MACP
h-sb	(cacic)	(acocic)
t-mn	(cacac)	(acocic)

%%% all other roots have Ø values for PACP and MACP.

c. A Dictionary of V-Affixes

c.i. A Dictionary of Sound Perfect V-Suffixes

	NBR	GDR	PRS	CAT	CAT2	MDG	VOC	MMK	REF
otu	sn	mf	r1	Ø	Ø	Ø	Ø	Ø	Ø
tu	sn	mf	r1	Ø	Ø	Ø	Ø	Ø	Ø
ona|	dp	mf	r1	Ø	Ø	Ø	Ø	Ø	Ø
na|	dp	mf	r1	Ø	GP	Ø	Ø	Ø	col
ota	sn	mc	r2	Ø	Ø	Ø	Ø	Ø	Ø
ta	sn	mc	r2	Ø	Ø	Ø	Ø	Ø	Ø
oti	sn	ff	r2	Ø	Ø	Ø	Ø	Ø	Ø
ti	sn	ff	r2	Ø	Ø	Ø	Ø	Ø	Ø
otuma|	dl	mf	r2	Ø	Ø	Ø	Ø	Ø	Ø
tuma|	dl	mf	r2	Ø	Ø	Ø	Ø	Ø	Ø
otumo	pl	mc	r2	Ø	Ø	Ø	Ø	Ø	Ø
tumo	pl	mc	r2	Ø	Ø	Ø	Ø	Ø	Ø
otunna	pl	ff	r2	Ø	Ø	Ø	Ø	Ø	Ø
tunna	pl	ff	r2	Ø	Ø	Ø	Ø	Ø	Ø
ona	pl	ff	r3	Ø	Ø	Ø	Ø	jis	Ø
na	pl	ff	r3	Ø	Ø	Ø	Ø	jis	Ø
a	sn	mc	r3	Ø	Ø	Ø	pas	sub	Ø
ato	sn	ff	r3	Ø	Ø	Ø	Ø	Ø	Ø
ata|	dl	ff	r3	Ø	Ø	Ø	Ø	Ø	Ø
a|	dl	mf	rx	Ø	Ø	Ø	Ø	sjj	Ø
uw|	pl	mc	r3	Ø	Ø	Ø	Ø	Ø	Ø

c.ii. A Dictionary of Defective Perfect V-Suffixes

	NBR	GDR	PRS	CAT	CAT2	MDG	VOC	MMK	REF
ay	Ø	Ø	Ø	Ø	Ø	Ø	pas	ids	Ø
awo|	pl	mc	r2	Ø	Ø	Ø	pas	sjj	Ø
awona	pl	mc	r2	Ø	Ø	Ø	Ø	Ø	Ø

c.iii. A Dictionary of Imperfect V-Prefixes

	NBR	GDR	PRS	CAT	CAT2	MDG	VOC	MMK	REF
:	sn	mf	r1	Ø	Ø	Ø	Ø	Ø	Ø
t	sn	mf	rx	Ø	Ø	Ø	Ø	Ø	Ø
y	sn	mc	r3	Ø	Ø	Ø	Ø	Ø	Ø
n	dp	mf	r1	Ø	Ø	Ø	Ø	Ø	Ø

c.iv. A Dictionary of Sound Imperfect V-Suffixes

	NBR	GDR	PRS	CAT	CAT2	MDG	VOC	MMK	REF
u	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ind	Ø
a	sn	mc	r3	Ø	Ø	Ø	pas	sub	Ø
o	Ø	Ø	Ø	Ø	Ø	Ø	Ø	jus	Ø
i	Ø	Ø	Ø	Ø	Ø	Ø	act	Ø	Ø
uwna	pl	mc	r2	Ø	Ø	Ø	act	ind	Ø
uw|	pl	mc	r3	Ø	Ø	Ø	act	sjj	Ø
iyna	sn	Ø	Ø	Ø	Ø	Ø	act	ind	Ø
iy	sn	mf	r1	Ø	GP	Ø	act	sjj	Ø
a|ni	dl	mf	rx	Ø	Ø	Ø	Ø	ind	Ø
a|	dl	mf	rx	Ø	Ø	Ø	Ø	sjj	Ø
ona	pl	ff	r3	Ø	Ø	Ø	Ø	jis	Ø
na	pl	ff	r3	Ø	Ø	Ø	Ø	jis	Ø

c.v. A Dictionary of Defective Imperfect V-Suffixes

	NBR	GDR	PRS	CAT	CAT2	MDG	VOC	MMK	REF
iy	sn	mf	r1	Ø	GP	Ø	act	sjj	Ø
ay	Ø	Ø	Ø	Ø	Ø	Ø	pas	ids	Ø
awo|	pl	mc	r2	Ø	Ø	Ø	pas	sjj	Ø
awona	pl	mc	r2	Ø	Ø	Ø	pas	ind	Ø

c.vi. A Dictionary of Bound Enclitic Pronouns

	NBR	GDR	PRS	HTY	DFN	GVT	CMK
niy	sn	mf	r1	Ø	dfp	Ø	Ø
iy	sn	mf	r1	hm	dfp	Ø	cpm
ya	sn	mf	r1	Ø	dfp	Ø	Ø
na|	dp	mf	r1	Ø	dfp	Ø	Ø
ka	sn	mc	r2	Ø	dfp	obm	Ø
ki	sn	ff	r2	Ø	dfp	Ø	Ø
kuma|	dl	mf	r2	Ø	dfp	Ø	
kumo	pl	mc	r2	Ø	dfp	Ø	Ø
kunna	pl	ff	r2	Ø	dfp	Ø	Ø
hu	sn	mc	r3	Ø	dfp	Ø	Ø
hi	sn	mc	r3	Ø	dfp	Ø	Ø
ha|	sn	ff	r3	Ø	dfp	Ø	
huma|	dl	mf	r3	Ø	dfp	Ø	nmm
hima|	dl	mf	r3	Ø	dfp	Ø	Ø
humo	pl	mc	r3	hm	dfp	Ø	nmm
himo	pl	mc	r3	Ø	dfp	Ø	Ø
hunna	pl	ff	r3	hm	dfp	Ø	nmm
hinna	pl	ff	r3	Ø	dfp	Ø	Ø

c.vi. A Dictionary of Bound Enclitic Pronouns (Contd)

	CMK2	FLXN	CAT	CAT2	TYP	PLC	REF
niy	obm	Ø	Ø	GP	Ø	Ø	col
iy	obm	dnf	Ø	GP	Ø	rmp	col
ya	obm	Ø	Ø	GP	Ø	Ø	col
na|	obm	Ø	Ø	GP	Ø	Ø	col
ka	obm	Ø	Ø	GP	Ø	Ø	col
ki	obm	Ø	Ø	GP	Ø	Ø	col
kuma|	obm	Ø	Ø	GP	Ø	Ø	col
kumo	obm	Ø	Ø	GP	Ø	Ø	col
kunna	obm	Ø	Ø	GP	Ø	Ø	col
hu	obm	Ø	Ø	GP	Ø	Ø	exl
hi	obm	Ø	Ø	GP	Ø	Ø	exl
ha|	obm	Ø	Ø	GP	Ø	Ø	exl
huma|	obm	Ø	Ø	GP	Ø	Ø	exl
hima|	obm	Ø	Ø	GP	Ø	Ø	exl
humo	obm	Ø	Ø	GP	Ø	Ø	exl
himo	obm	Ø	Ø	GP	Ø	Ø	exl
hunna	obm	Ø	Ø	GP	Ø	Ø	exl
hinna	obm	Ø	Ø	GP	Ø	Ø	exl

c.vii. A Dictionary of Future Particles

	NBR	GDR	PRS	CAT	CAT2	MDG	VOC	MMK	REF
sa	Ø	Ø	Ø	Ø	Ø	ind	Ø	Ø	Ø
li	Ø	Ø	Ø	Ø	Ø	sub	Ø	Ø	Ø

c.viii. A Dictionary of V-PATLISTS

	NBR	GDR	PRS	CAT	CAT2	MDG	VOC	MMK	REF
patlist1	Ø	Ø	Ø	Ø	Ø	Ø	act	Ø	Ø
patlist2	Ø	Ø	Ø	Ø	Ø	Ø	pas	Ø	Ø
patlist3	Ø	Ø	Ø	Ø	Ø	Ø	act	Ø	Ø
patlist4	Ø	Ø	Ø	Ø	Ø	Ø	pas	Ø	Ø
patlist5	Ø	Ø	Ø	Ø	Ø	Ø	act	Ø	Ø
patlist6	Ø	Ø	Ø	Ø	Ø	Ø	act	Ø	Ø

d. A Dictionary of Pronouns and Proper Nouns

d.i. A Dictionary of Subject Pronouns

	NBR	GDR	PRS	HTY	DFN	GVT	CMK
:ana|	sn	mf	r1	Ø	dfp	Ø	nmm
nah-onu	dp	mf	r1	hm	dfp	Ø	nmm
:anota	sn	mc	r2	Ø	dfp	Ø	nmm
:anoti	sn	ff	r2	Ø	dfp	Ø	nmm
:anotuma|	dl	mf	r2	Ø	dfp	Ø	nmm
:anotumo	pl	mc	r2	hm	dfp	Ø	nmm
:anotunna	pl	ff	r2	hm	dfp	Ø	nmm
huwa	sn	mc	r3	Ø	dfp	Ø	nmm
hiya	sn	ff	r3	Ø	dfp	Ø	nmm
huma|	dl	mf	r3	Ø	dfp	Ø	nmm
humo	pl	mc	r3	hm	dfp	Ø	nmm
hunna	pl	ff	r3	hm	dfp	Ø	nmm

d.i. A Dictionary of Subject Pronouns (Contd)

	CMK2	FLXN	CAT	CAT2	TYP	PLC	REF
:ana|	Ø	svm	Ø	Ø	Ø	Ø	Ø
nah-onu	Ø	svm	Ø	Ø	Ø	Ø	Ø
:anota	Ø	svm	Ø	Ø	Ø	Ø	Ø
:anoti	Ø	svm	Ø	Ø	Ø	Ø	Ø
:anotuma|	Ø	svm	Ø	Ø	Ø	Ø	Ø
:anotumo	Ø	svm	Ø	Ø	Ø	Ø	Ø
:anotunna	Ø	svm	Ø	Ø	Ø	Ø	Ø
huwa	Ø	svm	Ø	Ø	Ø	Ø	Ø
hiya	Ø	svm	Ø	Ø	Ø	Ø	Ø
huma|	obm	svm	Ø	GP	Ø	Ø	exl
humo	obm	svm	Ø	GP	Ø	Ø	exl
hunna	obm	svm	Ø	GP	Ø	Ø	exl

d.ii. A Dictionary of Demonstrative Pronouns

	NBR	GDR	PRS	HTY	DFN	GVT	CMK
had-a|	sn	mc	Ø	Ø	dfp	Ø	ncp
had-ihi	sn	ff	Ø	Ø	dfp	Ø	ncp
had-a|ni	dl	mc	Ø	Ø	dfp	Ø	nmm
had-ayoni	dl	mc	Ø	Ø	dfp	Ø	cpm
ha|ta|ni	dl	ff	Ø	Ø	dfp	Ø	nmm
ha|tayoni	dl	ff	Ø	Ø	dfp	Ø	cpm
ha:ula|:i	pl	mf	Ø	hm	dfp	Ø	ncp

d.ii. A Dictionary of Demonstrative Pronouns (Contd)

	CMK2	FLXN	CAT	CAT2	TYP	PLC	REF
had-a|	Ø	svm	Ø	Ø	Ø	Ø	Ø
had-ihi	Ø	svm	Ø	Ø	Ø	Ø	Ø
had-a|ni	Ø	svm	Ø	Ø	Ø	Ø	Ø
had-ayoni	Ø	svm	Ø	Ø	Ø	Ø	Ø
ha|ta|ni	Ø	svm	Ø	Ø	Ø	Ø	Ø
ha|tayoni	Ø	svm	Ø	Ø	Ø	Ø	Ø
ha:ula|:i	Ø	svm	Ø	Ø	Ø	Ø	Ø

d.iii. A Dictionary of Proper Nouns

	NBR	GDR	PRS	HTY	DFN	GVT	CMK	CMK2	FLXN
s;a|lih-	sn	mc	Ø	Ø	dfp	Ø	Ø	Ø	Ø
zayod	sn	mc	Ø	Ø	dfp	Ø	Ø	Ø	Ø
ɛamor	sn	mc	Ø	Ø	dfp	Ø	Ø	Ø	Ø
hisva|m	sn	mc	Ø	Ø	dfp	Ø	Ø	Ø	Ø
mlayek	sn	mc	Ø	Ø	dfp	Ø	Ø	Ø	Ø
ɛiysay	sn	mc	Ø	Ø	dfp	Ø	Ø	Ø	svm
muwsay	sn	mc	Ø	Ø	dfp	Ø	Ø	Ø	svm
yah-oyay	sn	mc	Ø	Ø	dfp	Ø	Ø	Ø	svm
riym	sn	ff	Ø	Ø	dfp	Ø	Ø	Ø	Ø
lat;iyfat	sn	ff	Ø	Ø	dfp	Ø	Ø	Ø	Ø
zakiyyat	sn	ff	Ø	Ø	dfp	Ø	Ø	Ø	Ø
ɛawa|t;if	sn	ff	Ø	Ø	dfp	Ø	Ø	Ø	Ø
zuhorat	sn	ff	Ø	Ø	dfp	Ø	Ø	Ø	Ø
d;uh-ay	sn	ff	Ø	Ø	dfp	Ø	Ø	Ø	svm
layolay	sn	ff	Ø	Ø	dfp	Ø	Ø	Ø	svm

B. THE L-DATABASE: A DICTIONARY OF L-PATTERNS

a. A Dictionary of L1-Patterns

	XDAP	MDAP	FDAP
ls	Ø	Ø	Ø
kn	(ca|:ic)	Ø	(ca|:ic cawa|:ic)
s;r	(ca|:ic)	Ø	Ø
bt	(ca|:ic)	Ø	Ø
bd;	(ca|:ic)	Ø	Ø
qm	Ø	(ca|:ic cuyya|c cuyyac)	(ca|:ic cawa|:ic)
t;l	(ca|:ic)	Ø	Ø
bb	Ø	Ø	Ø
jd	(ca|:ic)	Ø	Ø
xl	(ca|:ic)	Ø	Ø
s:	Ø	Ø	Ø
sd	(ca|:ic ca|cat)	Ø	Ø
bn	Ø	(ca|c cuca|t)	(ca|ciy cawa|c)
t-n	Ø	(ca|c)	(ca|ciy cawa|c)
d-k	Ø	(ca|c)	(ca|ciy)
m:	Ø	(ca|c)	(ca|ciy)
gn	Ø	(ca|c)	(ca|ciy cawa|c)
svq	Ø	(ca|c)	(ca|ciy cawa|c)
:b	Ø	Ø	Ø
:x	Ø	(muca|c)	(muca|ciy)
ns	Ø	(ca|c)	(ca|ciy cawa|c)
ld	(wa|cic)	Ø	Ø
rd	(wa|cic)	Ø	Ø

h-d	Ø	(ca|c)	(ca|ciy)
sm	(wa|cic)	Ø	Ø
:l	(ca|wic)	Ø	Ø
d/nn	(ca|cc)	Ø	Ø
jdd	(ca|cc)	Ø	Ø
ɛmm	(ca|cc cawa|cc)	Ø	Ø
h-sb	Ø	(ca|cic cacacat)	(ca|cic)
h-sn	(ca|cic)	Ø	Ø
krm	(ca|cic)	Ø	Ø
kbr	(ca|cic)	Ø	Ø
kt-r	(ca|cic)	Ø	Ø
s;gr	(ca|cic)	Ø	Ø
qs;r	Ø	(ca|cic)	(ca|cic cawa|cic)
jml	(ca|cic)	Ø	Ø
fqr	Ø	Ø	Ø
t-mn	Ø	(ca|cic)	(ca|cic cawa|cic)
svrf	Ø	(ca|cic cucuc cucuwc)	(ca|cic cawa|cic)
ktb	Ø	(ca|cic cucca|c cacacat)	(ca|cic)
svkr	Ø	(ca|cic cuccac)	(ca|cic)
drs	(ca|cic)	Ø	Ø
frsv	(ca|cic)	Ø	Ø
qtl	Ø	(ca|cic cucca|c cacacat)	(ca|cic)
rbɛ	Ø	(ca|cic)	(ca|cic cawa|cic)
skn	Ø	(ca|cic cucca|c)	(ca|cic cawa|cic)
fth-	Ø	(ca|cic)	(ca|cic cawa|cic)
mnh-	(ca|cic)	Ø	Ø
mnɛ	Ø	(ca|cic cacacat)	(ca|cic cawa|cic)
d-hb	Ø	(ca|cic)	(ca|cic cawa|cic)
xd;r	(ca|cic)	Ø	Ø
jls	Ø	(ca|cic cucuwc)	(ca|cic)
ksr	Ø	(ca|cic cuccac)	(ca|cic cawa|cic)
d;rb	Ø	(ca|cic)	(ca|cic cawa|cic)
nzl	Ø	(ca|cic)	(ca|cic cawa|cic)
h-ml	Ø	(ca|cic cacacat)	(ca|cic cawa|cic)
qlm	Ø	(ca|cic cacacat)	(ca|cic)
svbk	(ca|cic)	Ø	Ø
ɛsvr	Ø	(ca|cic)	(ca|cic cawa|cic)
sds	(ca|cic)	Ø	Ø
stt	Ø	Ø	Ø
sbɛ	Ø	(ca|cic cacacat)	(ca|cic cawa|cic)
t-lt-	Ø	(ca|cic)	(ca|cic cawa|cic)
tsɛ	(ca|cic)	Ø	Ø
xms	Ø	(ca|cic)	(ca|cic cawa|cic)
:lf	Ø	(ca|cic cucca|c)	(ca|cic cawa|cic)
s;fr	(ca|cic)	Ø	Ø
h-dq	(ca|cic)	Ø	Ø
fhm	(ca|cic)	Ø	Ø
ɛlm	Ø	(ca|cic cucca|c)	(ca|cic)
qdm	Ø	(ca|cic cucuc cucuwc)	(ca|cic cawa|cic)
svrb	Ø	(ca|cic cacoc)	(ca|cic cawa|cic)
rkb	Ø	(ca|cic cucca|c cucoca|n)	(ca|cic cawa|cic)
mr:	(ca|cic)	Ø	Ø
sɛd	Ø	Ø	Ø
rjl	Ø	(ca|cic cucca|c cucoca|n)	(ca|cic)
s;dq	(ca|cic)	Ø	Ø
nfd-	Ø	(ca|cic)	(ca|cic cawa|cic)

jbl	(ca|cic)	Ø	Ø
h-mr	(ca|cic)	Ø	Ø
zrq	(ca|cic)	Ø	Ø
:mm	(ca|cc)	Ø	Ø
kttb	(mucaccic)	Ø	Ø
drrs	(mucaccic)	Ø	Ø
frrsv	(mucaccic)	Ø	Ø
qttl	(mucaccic)	Ø	Ø
skkn	(mucaccic)	Ø	Ø
ftth-	(mucaccic)	Ø	Ø
jlls	(mucaccic)	Ø	Ø
kssr	(mucaccic)	Ø	Ø
d;rrb	(mucaccic)	Ø	Ø
nzzl	(mucaccic)	Ø	Ø
fhhm	(mucaccic)	Ø	Ø
ɛllm	(mucaccic)	Ø	Ø
svrrb	(mucaccic)	Ø	Ø
h-ssn	(mucaccic)	Ø	Ø
krrm	(mucaccic)	Ø	Ø
kbbr	(mucaccic)	Ø	Ø
kt-t-r	(mucaccic)	Ø	Ø
s;ggr	(mucaccic)	Ø	Ø
qs;s;r	(mucaccic)	Ø	Ø
jmml	(mucaccic)	Ø	Ø
fqqr	(mucaccic)	Ø	Ø
jddd	(mucaccic)	Ø	Ø
t-mmn	(mucaccic)	Ø	Ø
svrrf	(mucaccic)	Ø	Ø
rbbɛ	(mucaccic)	Ø	Ø
h-mml	(mucaccic)	Ø	Ø
qllm	(mucaccic)	Ø	Ø
svbbk	(mucaccic)	Ø	Ø
ɛsvsvr	(mucaccic)	Ø	Ø
sdds	(mucaccic)	Ø	Ø
sbbɛ	(mucaccic)	Ø	Ø
t-llt-	(mucaccic)	Ø	Ø
xmms	(mucaccic)	Ø	Ø
:llf	(mucaccic)	Ø	Ø
s;ffr	(mucaccic)	Ø	Ø
rkkb	(mucaccic)	Ø	Ø
mrr:	(mucaccic)	Ø	Ø
rjjl	(mucaccic)	Ø	Ø
s;ddq	(mucaccic)	Ø	Ø
nffd-	(mucaccic)	Ø	Ø
h-ddq	(mucaccic)	Ø	Ø
jbbl	(mucaccic)	Ø	Ø
xd;d;r	(mucaccic)	Ø	Ø
h-mmr	(mucaccic)	Ø	Ø
:mmm	(mucaccic)	Ø	Ø
ɛmmm	(mucaccic)	Ø	Ø
bnn	(mucacc)	(mucacciy)	Ø
t-nn	(mucacc)	(mucacciy)	Ø
d-kk	(mucacc)	(mucacciy)	Ø
gnn	(mucacc)	(mucacciy)	Ø
ssm	(muwaccic)	Ø	Ø
lld	(muwaccic)	Ø	Ø

rrd	(muwaccic)	Ø	Ø
h-h-d	(muwaccic)	Ø	Ø
s;yyr	(mucayyic)	Ø	Ø
byyt	(mucayyic)	Ø	Ø
byyd;	(mucayyic)	Ø	Ø
:wwl	(mucawwic)	Ø	Ø
kwwn	(mucawwic)	Ø	Ø
qwwm	(mucawwic)	Ø	Ø
t;wwl	(mucawwic)	Ø	Ø
bwwb	(mucawwic)	Ø	Ø
jwwd	(mucawwic)	Ø	Ø
xwwl	(mucawwic)	Ø	Ø
sww:	(mucawwic)	Ø	Ø
swwd	(mucawwic)	Ø	Ø

b. A Dictionary of L2-Patterns

	XDSP	MDSP	FDSP
ls	Ø	Ø	Ø
kn	(macuwc)	Ø	Ø
s;r	Ø	Ø	Ø
bt	Ø	Ø	Ø
bd;	Ø	Ø	Ø
qm	(macuwc)	Ø	Ø
t;l	Ø	Ø	Ø
bb	Ø	Ø	Ø
jd	Ø	Ø	Ø
xl	Ø	Ø	Ø
s:	Ø	Ø	Ø
sd	(macuwc)	Ø	Ø
bn	(macociyy)	Ø	Ø
t-n	(macociyy)	Ø	Ø
d-k	Ø	Ø	Ø
m:	(macociyy)	Ø	Ø
gn	Ø	Ø	Ø
svq	Ø	Ø	Ø
:b	Ø	Ø	Ø
:x	Ø	(muca|cay)	(muca|ca|t muca|cay)
ns	(macociyy)	Ø	Ø
ld	(mawocuwc)	Ø	Ø
rd	(mawocuwc)	Ø	Ø
h-d	(mawocuwc)	Ø	Ø
sm	(mawocuwc)	Ø	Ø
:l	Ø	Ø	Ø
d/nn	(macocuwc)	Ø	Ø
jdd	Ø	Ø	Ø
ɛmm	Ø	Ø	Ø
h-sb	(macocuwc)	Ø	Ø
h-sn	Ø	Ø	Ø
krm	Ø	Ø	Ø
kbr	Ø	Ø	Ø
kt-r	Ø	Ø	Ø
s;gr	Ø	Ø	Ø
qs;r	Ø	Ø	Ø
jml	Ø	Ø	Ø
fqr	Ø	Ø	Ø
t-mn	(macocuwc)	Ø	Ø

svrf	Ø	Ø	Ø
ktb	(macocuwc)	Ø	Ø
svkr	(macocuwc)	Ø	Ø
drs	(macocuwc)	Ø	Ø
frsv	(macocuwc)	Ø	Ø
qtl	(macocuwc)	Ø	Ø
rbɛ	(macocuwc)	Ø	Ø
skn	(macocuwc)	Ø	Ø
fth-	(macocuwc)	Ø	Ø
mnh-	(macocuwc)	Ø	Ø
mnɛ	(macocuwc)	Ø	Ø
d-hb	(macocuwc)	Ø	Ø
xd;r	Ø	Ø	Ø
jls	(macocuwc)	Ø	Ø
ksr	(macocuwc)	Ø	Ø
d;rb	(macocuwc)	Ø	Ø
nzl	(macocuwc)	Ø	Ø
h-ml	(macocuwc)	Ø	Ø
qlm	(macocuwc)	Ø	Ø
svbk	(macocuwc)	Ø	Ø
ɛsvr	(macocuwc)	Ø	Ø
sds	(macocuwc)	Ø	Ø
stt	Ø	Ø	Ø
sbɛ	(macocuwc)	Ø	Ø
t-lt-	(macocuwc)	Ø	Ø
tsɛ	(macocuwc)	Ø	Ø
xms	(macocuwc)	Ø	Ø
:lf	(macocuwc)	Ø	Ø
s;fr	Ø	Ø	Ø
h-dq	(macocuwc)	Ø	Ø
fhm	(macocuwc)	Ø	Ø
ɛlm	(macocuwc)	Ø	Ø
qdm	(macocuwc)	Ø	Ø
svrb	(macocuwc)	Ø	Ø
rkb	(macocuwc)	Ø	Ø
mr:	(macocuwc)	Ø	Ø
sɛd	Ø	Ø	Ø
rjl	Ø	Ø	Ø
s;dq	Ø	Ø	Ø
nfd-	Ø	Ø	Ø
jbl	(macocuwc)	Ø	Ø
h-mr	Ø	Ø	Ø
zrq	Ø	Ø	Ø
:mm	Ø	Ø	Ø
kttb	(mucaccac)	Ø	Ø
drrs	(mucaccac)	Ø	Ø
frrsv	(mucaccac)	Ø	Ø
qttl	(mucaccac)	Ø	Ø
skkn	(mucaccac)	Ø	Ø
ftth-	(mucaccac)	Ø	Ø
jlls	(mucaccac)	Ø	Ø
kssr	(mucaccac)	Ø	Ø
d;rrb	(mucaccac)	Ø	Ø
nzzl	(mucaccac)	Ø	Ø
fhhm	(mucaccac)	Ø	Ø

ɛllm	(mucaccac)	Ø	Ø
svrrb	(mucaccac)	Ø	Ø
h-ssn	(mucaccac)	Ø	Ø
krrm	(mucaccac)	Ø	Ø
kbbr	(mucaccac)	Ø	Ø
kt-t-r	(mucaccac)	Ø	Ø
s;ggr	(mucaccac)	Ø	Ø
qs;s;r	(mucaccac)	Ø	Ø
jmml	(mucaccac)	Ø	Ø
fqqr	(mucaccac)	Ø	Ø
jddd	(mucaccac)	Ø	Ø
t-mmn	(mucaccac)	Ø	Ø
svrrf	(mucaccac)	Ø	Ø
rbbɛ	(mucaccac)	Ø	Ø
h-mml	(mucaccac)	Ø	Ø
qllm	(mucaccac)	Ø	Ø
svbbk	(mucaccac)	Ø	Ø
ɛsvsvr	(mucaccac)	Ø	Ø
sdds	(mucaccac)	Ø	Ø
sbbɛ	(mucaccac)	Ø	Ø
t-llt-	(mucaccac)	Ø	Ø
xmms	(mucaccac)	Ø	Ø
:llf	(mucaccac)	Ø	Ø
s;ffr	(mucaccac)	Ø	Ø
rkkb	(mucaccac)	Ø	Ø
mrr:	(mucaccac)	Ø	Ø
rjjl	(mucaccac)	Ø	Ø
s;ddq	(mucaccac)	Ø	Ø
nffd-	(mucaccac)	Ø	Ø
h-ddq	(mucaccac)	Ø	Ø
jbbl	(mucaccac)	Ø	Ø
xd;d;r	(mucaccac)	Ø	Ø
h-mmr	(mucaccac)	Ø	Ø
:mmm	(mucaccac)	Ø	Ø
ɛmmm	(mucaccac)	Ø	Ø
bnn	Ø	(mucaccay)	(mucacca|t mucaccay)
t-nn	Ø	(mucaccay)	(mucacca|t mucaccay)
d-kk	Ø	(mucaccay)	(mucacca|t mucaccay)
gnn	Ø	(mucaccay)	(mucacca|t mucaccay)
ssm	(muwaccac)	Ø	Ø
lld	(muwaccac)	Ø	Ø
rrd	(muwaccac)	Ø	Ø
h-h-d	(muwaccac)	Ø	Ø
s;yyr	(mucayyac)	Ø	Ø
byyt	(mucayyac)	Ø	Ø
byyd;	(mucayyac)	Ø	Ø
:wwl	(mucawwac)	Ø	Ø
kwwn	(mucawwac)	Ø	Ø
qwwm	(mucawwac)	Ø	Ø
t;wwl	(mucawwac)	Ø	Ø
bwwb	(mucawwac)	Ø	Ø
jwwd	(mucawwac)	Ø	Ø
xwwl	(mucawwac)	Ø	Ø
sww:	(mucawwac)	Ø	Ø
swwd	(mucawwac)	Ø	Ø

C. THE S-DATABASE

a. A Dictionary of Noun Patterns

a.i. The MF-Patterns

	XNNP
ls	Ø
kn	Ø
s;r	Ø
bt	Ø
bd;	Ø
qm	Ø
t;l	Ø
bb	Ø
jd	Ø
xl	Ø
s:	Ø
sd	Ø
bn	(cica|: :acociyat)
t-n	Ø
d-k	Ø
m:	Ø
gn	Ø
svq	Ø
:b	Ø
:x	Ø
ns	Ø
ld	Ø
rd	Ø
h-d	Ø
sm	Ø
:l	Ø
d/nn	Ø
jdd	Ø
ɛmm	Ø
h-sb	Ø
h-sn	Ø
krm	Ø
kbr	Ø
kt-r	Ø
s;gr	Ø
qs;r	Ø
jml	Ø
fqr	Ø
t-mn	Ø
svrf	Ø
ktb	(caciyc caca|:ic)
svkr	Ø
drs	Ø
frsv	Ø
qtl	Ø
rbɛ	Ø
skn	Ø
fth-	Ø
mnh-	(caciyc caca|:ic)
mnɛ	Ø

d-hb	(cacac :acoca|c cica|c)
xd;r	Ø
jls	Ø
ksr	Ø
d;rb	Ø
nzl	Ø
h-ml	(caciyc caca|:ic)
qlm	Ø
svbk	Ø
ɛsvr	Ø
sds	Ø
stt	Ø
sbɛ	Ø
t-lt-	Ø
tsɛ	Ø
xms	Ø
:lf	Ø
s;fr	Ø
h-dq	Ø
fhm	Ø
ɛlm	Ø
qdm	Ø
svrb	Ø
rkb	(cica|c caca|:ic)
mr:	(cacoc)
sɛd	(ca|cic cawa|cic)
rjl	Ø
s;dq	Ø
nfd-	Ø
jbl	Ø
h-mr	(caciyc caca|:ic)
zrq	Ø
:mm	(cica|c :aciccat cayiccat)

a.ii. The M- and F-Patterns

	MNNP	FNNP
ls	Ø	Ø
kn	(cawoc :acowa|c)	Ø
s;r	Ø	Ø
bt	(cayoc :acoya|c cuyuwc)	Ø
bd;	Ø	(cayoc cuyuwc cayoc)
qm	(cawoc :acowa|c)	(ca|c ciyac)
t;l	Ø	Ø
bb	(ca|c :acowa|c :acowicat)	(ca|c)
jd	(cawa|c :acoya|c ciya|c)	Ø
xl	(ca|c :acowa|c :acowicat)	(ca|c)
s:	Ø	Ø
sd	(cayyic :acoya|c ca|cat)	(cayyic)
bn	(:icoc :acoca|:)	(cic)
t-n	Ø	Ø
d-k	Ø	Ø
m:	Ø	Ø
gn	Ø	Ø
svq	Ø	Ø
:b	(cac ca|ca|:)	Ø
:x	(cac ca|ca|: cicowat)	Ø

ns	Ø	(Ø cica|: cicowat)
ld	(wacac :awoca|c wicocat)	(waciyc waca|:ic)
rd	(waciyc :awocicat)	(wacoc wucuwc :awoca|c)
h-d	Ø	(wicoc)
sm	(wica|c :awocicat)	Ø
:l	Ø	Ø
d/nn	(caciyc :acicca|:)	(caciyc)
jdd	(cacc :acoca|c cucuwcat)	(cacc)
ɛmm	(cacc :acoca|c cucuwcat)	(cacc)
h-sb	(cica|c cucoca|n)	Ø
h-sn	Ø	Ø
krm	Ø	Ø
kbr	Ø	Ø
kt-r	Ø	Ø
s;gr	Ø	Ø
qs;r	(cacoc cucuwc)	(cacac :acoca|c cacac)
jml	(cacac :acoca|c cica|c)	Ø
fqr	Ø	Ø
t-mn	(cacac :acoca|c :acocicat)	Ø
svrf	Ø	(cucoc cucac)
ktb	(cica|c cucuc)	(cica|c)
svkr	(caciyc cucuc)	Ø
drs	(cacoc cucuwc)	(cica|c)
frsv	(cica|c cucuc)	(caciyc caca|:ic)
qtl	Ø	Ø
rbɛ	(caciyc cucuc cica|c)	(caciyc caca|:ic)
skn	Ø	(caciyc caca|:ic)
fth-	(micoca|c maca|ciyc)	Ø
mnh-	Ø	(cicoc cicac)
mnɛ	Ø	Ø
d-hb	Ø	Ø
xd;r	Ø	Ø
jls	(caciyc cucaca|: cucca|c)	(caciyc)
ksr	Ø	Ø
d;rb	(caciyc cucaca|:)	(caciyc caca|:ic)
nzl	(caciyc cucaca|:)	(caciyc)
h-ml	Ø	Ø
qlm	(cacac :acoca|c cica|c)	Ø
svbk	(cucca|c caca|ciyc)	(cacac cica|c)
ɛsvr	(caciyc :acocica|: cucaca|:)	(caciyc caca|:ic)
sds	(caciyc :acoca|c)	Ø
stt	Ø	Ø
sbɛ	(caciyc :acoca|c)	(cacuc)
t-lt-	(caciyc :acoca|c)	Ø
tsɛ	(caciyc cucuc)	Ø
xms	(caciyc :acocica|: :acocicat)	Ø
:lf	(caciyc cucaca|:)	(caciyc caca|:ic)
s;fr	(caciyc)	Ø
h-dq	Ø	(cacac :acoca|c cacac)
fhm	Ø	Ø
ɛlm	(cacac :acoca|c cica|c)	(cacac)
qdm	Ø	(cacac :acoca|c cuca|c)
svrb	(caca|c :acocicat)	Ø
rkb	(caciyc cucuc)	(caciyc)
mr:	Ø	(:icocac)
sɛd	Ø	Ø

rjl	(cacuc :aca|cic cica|c)	Ø
s;dq	(caciyc :acocica|: cucaca|:)	(caciyc)
nfd-	Ø	Ø
jbl	(caciyc cucuc)	(caciyc caca|:ic)
h-mr	(cica|c cucuc :acocicat)	(cica|c)
zrq	Ø	Ø
:mm	Ø	(cucc cucac)

%%% in addition stems: {bn, ld, h-sb, frsv, drs, sbε, h-dq, s;dq, jbl, :mm} have these patterns:

	MNNP2	FNNP2
bn	Ø	(:icoc)
ld	(waciyc wicoca|n)	(wica|c)
h-sb	(cacac :acoca|c)	Ø
frsv	Ø	(caca|c)
drs	(caciyc :acoca|c cicoca|n)	Ø
sbε	(cacuc cucuwc cica|c)	Ø
h-dq	Ø	(caciyc caca|:ic)
s;dq	Ø	(caca|c)
jbl	(cacac :acoca|c cica|c)	Ø
:mm	Ø	(cuccah)

%%% all other stem have Ø values for MNNP2 and FNNP2.

b. A Dictionary of N-Affixes

b.i. A Dictionary of Nominal CASE-only Suffixes

	NBR	GDR	PRS	HTY	DFN	GVT	CMK	CMK2	FLXN
o	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	dnf
a	sn	mc	r3	Ø	Ø	Ø	acm	Ø	svm
u	Ø	Ø	Ø	Ø	Ø	Ø	nmm	Ø	svm
i	Ø	Ø	Ø	Ø	Ø	Ø	obm	Ø	svm
a+	Ø	Ø	Ø	Ø	Ø	Ø	acm	Ø	fvm
u+	Ø	Ø	Ø	Ø	Ø	Ø	nmm	Ø	fvm
i+	Ø	Ø	Ø	Ø	Ø	Ø	obm	Ø	fvm

b.ii. A Dictionary of Nominal GENDER-only Suffixes

	NBR	GDR	PRS	HTY	DFN	GVT	CMK	CMK2	FLXN	PLC
ot	sn	ff	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
at	sn	ff	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
a|t	pl	ff	Ø	hm	Ø	Ø	Ø	Ø	Ø	rfp

b.iii. A Dictionary of Nominal NGC Suffixes

	NBR	GDR	PRS	HTY	DFN	CMK	CMK2	FLXN	CAT2	PLC	REF
a|ni	dl	mf	rx	Ø	Ø	nmm	Ø	nnf	Ø	Ø	Ø
a|	dl	mf	rx	Ø	Ø	nmm	Ø	dnf	Ø	Ø	Ø
ayoni	dl	Ø	Ø	Ø	Ø	cpm	Ø	nnf	Ø	Ø	Ø
ayo	dl	Ø	Ø	Ø	Ø	cpm	Ø	dnf	Ø	Ø	Ø
uwna	pl	mc	r2	hm	Ø	nmm	Ø	nnf	Ø	rmp	Ø
awona	pl	mc	r2	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
uw	pl	mc	Ø	hm	Ø	nmm	Ø	dnf	Ø	rmp	Ø
iyna	sn	Ø	Ø	hm	Ø	cpm	Ø	nnf	Ø	rmp	Ø
iy	sn	mf	r1	hm	dfp	cpm	obm	dnf	GP	rmp	col
ay	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

b.iv. A Dictionary of Bound Prepositions

	NBR	GDR	PRS	DFN	GVT	CMK	CMK2	CAT2	TYP	REF
li	Ø	Ø	Ø	Ø	obm	Ø	Ø	Ø	Ø	Ø
la	Ø	Ø	Ø	Ø	obm	Ø	Ø	Ø	Ø	Ø
l	Ø	Ø	Ø	Ø	obm	Ø	Ø	Ø	bpr	Ø
bi	Ø	Ø	Ø	Ø	obm	Ø	Ø	Ø	Ø	Ø
b	Ø	Ø	Ø	Ø	obm	Ø	Ø	Ø	Ø	Ø
ka	sn	mc	r2	dfp	obm	Ø	obm	GP	Ø	col

b.v. The Determiner

	GVT	CMK	CMK2	FLXN	CAT	CAT2	TYP	PLC	REF
:alo	Ø	Ø	Ø	Ø	Ø	Ø	nbd	Ø	Ø
:al	Ø	Ø	Ø	Ø	Ø	Ø	nbd	Ø	Ø
lo	Ø	Ø	Ø	Ø	Ø	Ø	bpr	Ø	Ø
l	obm	Ø	Ø	Ø	Ø	Ø	bpr	Ø	Ø

b.vi. NUMERICAL VALUES

(h-d 1)	(t-n 2)	(t-lt- 3)
(rbɛ 4)	(xms 5)	(stt 6)
(sbɛ 7)	(t-mn 8)	(tsɛ 9)
(ɛsvr 10)	(m: 100)	(:lf 1000)

b.vii. A Dictionary of N-PATLISTS

	NBR	GDR	PRS	CMK	CMK2	FLXN	CAT	CAT2	PLC
NNLIST									
nnlist1	Ø	Ø	Ø	Ø	Ø	Ø	NN	Ø	Ø
nnlist2	Ø	Ø	Ø	Ø	Ø	Ø	NN	Ø	Ø
nnlistb	Ø	Ø	Ø	Ø	Ø	Ø	NN	Ø	Ø
nnlist3	Ø	Ø	Ø	Ø	Ø	Ø	NN	Ø	Ø
nnlistc	Ø	Ø	Ø	Ø	Ø	Ø	NN	Ø	Ø
MLIST+									
dplist1	Ø	Ø	Ø	Ø	Ø	Ø	L1	act	Ø
dplist4	Ø	Ø	Ø	Ø	Ø	Ø	L2	pas	Ø
dplist5	Ø	Ø	Ø	Ø	Ø	Ø	L2	pas	Ø
!*FLIST									
dplist1	Ø	Ø	Ø	Ø	Ø	Ø	L1	act	Ø
dplist3	Ø	Ø	Ø	Ø	Ø	Ø	L1	act	Ø
dplist4	Ø	Ø	Ø	Ø	Ø	Ø	L2	pas	Ø
CVLIST									
cvlist1	Ø	Ø	Ø	Ø	Ø	Ø	NN	Ø	Ø
cvlist2	Ø	Ø	Ø	Ø	Ø	Ø	NN	Ø	Ø
cvlist3	Ø	Ø	Ø	Ø	Ø	Ø	NN	Ø	Ø
ADLIST									
adlist1	Ø	Ø	Ø	Ø	Ø	Ø	AA	Ø	Ø
adlist2	Ø	Ø	Ø	Ø	Ø	Ø	AA	Ø	Ø
adlist3	Ø	Ø	Ø	Ø	Ø	Ø	AA	Ø	Ø
MMLIST									
mmlist1	Ø	Ø	Ø	Ø	Ø	Ø	MM	Ø	Ø
mmlist2	Ø	Ø	Ø	Ø	Ø	Ø	MM	Ø	Ø
mmlist3	Ø	Ø	Ø	Ø	Ø	Ø	MM	Ø	Ø
mmlist4	Ø	Ø	Ø	Ø	Ø	Ø	MM	Ø	Ø

D. THE C-DATABASE: A DICTIONARY OF TLOCATIVE PATTERNS

	XLOCVP	MLOCVP	FLOCVP
ls	Ø	Ø	Ø
kn	Ø	(maca|c :ama|cic :amocicat)	(maca|c)
s;r	Ø	(maciyc maca|yic)	Ø
bt	Ø	(maciyc)	Ø
bd;	Ø	Ø	Ø
qm	(maca|c)	Ø	Ø
t;l	Ø	Ø	Ø
bb	Ø	Ø	Ø
jd	Ø	Ø	Ø
xl	Ø	Ø	Ø
s:	Ø	Ø	Ø
sd	Ø	Ø	Ø
bn	Ø	(macocay maca|c)	Ø
t-n	Ø	(macocay maca|c)	(maca|c)
d-k	Ø	Ø	Ø
m:	Ø	Ø	Ø
gn	Ø	(macocay maca|c)	Ø
svq	Ø	Ø	Ø
:b	Ø	Ø	Ø
:x	Ø	Ø	Ø
ns	Ø	Ø	Ø
ld	Ø	(mawocic mawa|cic)	Ø
rd	Ø	(mawocic mawa|cic)	Ø
h-d	Ø	Ø	Ø
sm	Ø	(mawocic mawa|cic)	Ø
:l	Ø	Ø	Ø
d/nn	Ø	Ø	(macicc maca|cc)
jdd	Ø	Ø	Ø
ɛmm	Ø	Ø	Ø
h-sb	Ø	Ø	(macocic maca|cic)
h-sn	Ø	(macocac maca|cic)	Ø
krm	Ø	Ø	Ø
kbr	Ø	Ø	Ø
kt-r	Ø	Ø	Ø
s;gr	Ø	Ø	Ø
qs;r	(macocac maca|cic)	Ø	Ø
jml	Ø	Ø	Ø
fqr	Ø	Ø	Ø
t-mn	Ø	(micocac maca|cic)	(micocac maca|cic)
svrf	Ø	(macocac maca|cic)	Ø
ktb	(macocac maca|cic)	Ø	Ø
svkr	Ø	Ø	Ø
drs	(macocac maca|cic)	(micocac maca|cic)	Ø
frsv	(micocac maca|cic)	Ø	Ø
qtl	Ø	(macocac maca|cic)	Ø
rbɛ	(macocac maca|cic)	Ø	Ø
skn	Ø	(macocac maca|cic)	Ø
fth-	Ø	(macocac maca|cic)	Ø
mnh-	Ø	Ø	Ø
mnɛ	Ø	Ø	Ø
d-hb	Ø	(macocac maca|cic)	Ø
xd;r	Ø	(micocac maca|cic)	(macocac)
jls	Ø	(macocic maca|cic)	Ø

ksr	Ø	(macocic maca|cic)	Ø
d;rb	(macocic maca|cic)	Ø	Ø
nzl	(macocic maca|cic)	Ø	Ø
h-ml	Ø	(macocic maca|cic)	Ø
qlm	Ø	(micocac maca|cic)	Ø
svbk	Ø	Ø	Ø
ɛsvr	Ø	(macocac maca|cic)	Ø
sds	Ø	Ø	Ø
stt	Ø	Ø	Ø
sbɛ	(macocac maca|cic)	Ø	Ø
t-lt-	Ø	(micocac maca|cic)	Ø
tsɛ	Ø	Ø	Ø
xms	Ø	Ø	Ø
:lf	Ø	Ø	Ø
s;fr	Ø	Ø	Ø
h-dq	Ø	Ø	Ø
fhm	Ø	Ø	Ø
ɛlm	Ø	(macocac maca|cic)	Ø
qdm	Ø	(macocac maca|cic)	Ø
svrb	(macocac maca|cic)	Ø	(micocac maca|cic)
rkb	(macocac maca|cic)	Ø	Ø
mr:	Ø	Ø	Ø
sɛd	Ø	Ø	Ø
rjl	Ø	(micocac maca|cic)	Ø
s;dq	Ø	Ø	Ø
nfd-	Ø	(macocic maca|cic)	Ø
jbl	Ø	(macocac maca|cic)	Ø
h-mr	Ø	(micocac maca|cic)	Ø
zrq	Ø	Ø	Ø
:mm	Ø	Ø	Ø

E. THE A-DATABASE: A DICTIONARY OF ADJECTIVE PATTERNS

	XADJP	MADJP	FADJP
ls	Ø	Ø	Ø
kn	Ø	Ø	Ø
s;r	Ø	Ø	Ø
bt	Ø	Ø	Ø
bd;	Ø	(:acoyac ciyc)	(cayoca|:)
qm	(cawiyc ciya|c)	Ø	Ø
t;l	Ø	(cawiyc ciwa|c ciya|c)	(cawiyc)
bb	Ø	Ø	Ø
jd	Ø	(cayyic caya|:ic)	(cayyic)
xl	Ø	Ø	Ø
s:	(cayyic)	Ø	Ø
sd	Ø	(:acowac cuwc)	(cawoca|:)
bn	Ø	Ø	Ø
t-n	Ø	Ø	Ø
d-k	Ø	(caciyy :acociya|:)	(caciyy)
m:	Ø	Ø	Ø
gn	Ø	(caciyy :acociya|:)	(caciyy)
svq	Ø	(caciyy :acociya|:)	(caciyy)
:b	Ø	Ø	Ø
:x	Ø	Ø	Ø
ns	Ø	Ø	Ø

ld	Ø	Ø	Ø
rd	Ø	Ø	Ø
h-d	(waciyc)	Ø	Ø
sm	(waciyc)	(waciyc wucaca|:)	Ø
:l	Ø	(cawwac cawa|:ic)	(cuwcay cuwac)
d/nn	Ø	Ø	Ø
jdd	Ø	(caciyc cucuc)	(caciyc)
ɛmm	Ø	Ø	Ø
h-sb	Ø	(caciyc cucaca|:)	(caciyc)
h-sn	(cacac cica|c)	Ø	Ø
krm	Ø	(caciyc cucaca|: cica|c)	(caciyc caca|:ic)
kbr	Ø	(caciyc cucaca|: cica|c)	(caciyc caca|:ic)
kt-r	(caciyc)	Ø	Ø
s;gr	Ø	(caciyc cucaca|: cica|c)	(caciyc caca|:ic)
qs;r	(caciyc cica|c)	(caciyc cucaca|:)	(caciyc caca|:ic)
jml	Ø	(caciyc cucaca|:)	(caciyc caca|:ic)
fqr	Ø	(caciyc cucaca|:)	(caciyc caca|:ic)
t-mn	Ø	(caciyc cucuc)	(caciyc)
svrf	Ø	(caciyc cucaca|: :acoca|c)	(caciyc caca|:ic)
ktb	Ø	Ø	Ø
svkr	Ø	Ø	Ø
drs	Ø	Ø	Ø
frsv	Ø	Ø	Ø
qtl	Ø	(caciyc cucaca|: cuca|cay)	(caciyc)
rbɛ	Ø	Ø	Ø
skn	Ø	Ø	Ø
fth-	Ø	Ø	Ø
mnh-	Ø	Ø	Ø
mnɛ	(caciyc caca|:ic)	Ø	Ø
d-hb	(caciyc caca|:ic)	Ø	Ø
xd;r	Ø	(:acocac cucoc)	(cacoca|:)
jls	Ø	Ø	Ø
ksr	Ø	(caciyc cacocay caca|cay)	(caciyc)
d;rb	Ø	Ø	Ø
nzl	Ø	Ø	Ø
h-ml	Ø	Ø	Ø
qlm	Ø	Ø	Ø
svbk	Ø	Ø	Ø
ɛsvr	Ø	Ø	Ø
sds	Ø	Ø	Ø
stt	Ø	Ø	Ø
sbɛ	Ø	Ø	Ø
t-lt-	Ø	Ø	Ø
tsɛ	Ø	Ø	Ø
xms	Ø	Ø	Ø
:lf	Ø	Ø	Ø
s;fr	Ø	(:acocac cucoc)	(cacoca|:)
h-dq	Ø	Ø	Ø
fhm	Ø	Ø	Ø
ɛlm	Ø	(caciyc cucaca|:)	(caciyc)
qdm	(caciyc caca|:ic)	(caciyc cucaca|: cuca|cay)	Ø
svrb	Ø	(caciyc cucaca|: cica|c)	(caciyc)
rkb	Ø	Ø	Ø
mr:	Ø	(caciyc cucuwc :acocicat)	(caciyc)
sɛd	Ø	(caciyc cucaca|:)	(caciyc)
rjl	Ø	(caciyc cacocay caca|cay)	(caciyc)

s;dq	Ø	Ø	Ø
nfd-	Ø	(caciyc cucuc)	(caciyc)
jbl	Ø	Ø	Ø
h-mr	Ø	(:acocac cucoc)	(cacoca|:)
zrq	Ø	(:acocac cucoc)	(cacoca|:)
:mm	(caciyc caca|:ic)	Ø	Ø

F. THE M-DATABASE: A DICTIONARY OF NUMERAL PATTERNS

	XNUMP	MNUMP	FNUMP	DNUMP
ls	Ø	Ø	Ø	Ø
kn	Ø	Ø	Ø	Ø
s;r	Ø	Ø	Ø	Ø
bt	Ø	Ø	Ø	Ø
bd;	Ø	Ø	Ø	Ø
qm	Ø	Ø	Ø	Ø
t;l	Ø	Ø	Ø	Ø
bb	Ø	Ø	Ø	Ø
jd	Ø	Ø	Ø	Ø
xl	Ø	Ø	Ø	Ø
s:	Ø	Ø	Ø	Ø
sd	Ø	Ø	Ø	Ø
bn	Ø	Ø	Ø	Ø
t-n	Ø	Ø	Ø	(:icoc)
d-k	Ø	Ø	Ø	Ø
m:	Ø	Ø	(cic)	Ø
gn	Ø	Ø	Ø	Ø
svq	Ø	Ø	Ø	Ø
:b	Ø	Ø	Ø	Ø
:x	Ø	Ø	Ø	Ø
ns	Ø	Ø	Ø	Ø
ld	Ø	Ø	Ø	Ø
rd	Ø	Ø	Ø	Ø
h-d	(wa|cic)	(:acac ca|ca|c)	(:icocay)	Ø
sm	Ø	Ø	Ø	Ø
:l	Ø	Ø	Ø	Ø
d/nn	Ø	Ø	Ø	Ø
jdd	Ø	Ø	Ø	Ø
ɛmm	Ø	Ø	Ø	Ø
h-sb	Ø	Ø	Ø	Ø
h-sn	Ø	Ø	Ø	Ø
krm	Ø	Ø	Ø	Ø
kbr	Ø	Ø	Ø	Ø
kt-r	Ø	Ø	Ø	Ø
s;gr	Ø	Ø	Ø	Ø
qs;r	Ø	Ø	Ø	Ø
jml	Ø	Ø	Ø	Ø
fqr	Ø	Ø	Ø	Ø
t-mn	Ø	(caca|c)	(caca|ciy)	Ø
svrf	Ø	Ø	Ø	Ø
ktb	Ø	Ø	Ø	Ø
svkr	Ø	Ø	Ø	Ø
drs	Ø	Ø	Ø	Ø
frsv	Ø	Ø	Ø	Ø
qtl	Ø	Ø	Ø	Ø

rbɛ	(:acocac)	Ø	Ø	Ø
skn	Ø	Ø	Ø	Ø
fth-	Ø	Ø	Ø	Ø
mnh-	Ø	Ø	Ø	Ø
mnɛ	Ø	Ø	Ø	Ø
d-hb	Ø	Ø	Ø	Ø
xd;r	Ø	Ø	Ø	Ø
jls	Ø	Ø	Ø	Ø
ksr	Ø	Ø	Ø	Ø
d;rb	Ø	Ø	Ø	Ø
nzl	Ø	Ø	Ø	Ø
h-ml	Ø	Ø	Ø	Ø
qlm	Ø	Ø	Ø	Ø
svbk	Ø	Ø	Ø	Ø
ɛsvr	Ø	(cacoc)	(cacac)	Ø
sds	Ø	Ø	Ø	Ø
stt	(cicc)	Ø	Ø	Ø
sbɛ	(cacoc)	Ø	Ø	Ø
t-lt-	(caca|c)	Ø	Ø	Ø
tsɛ	(cicoc)	Ø	Ø	Ø
xms	(cacoc)	Ø	Ø	Ø
:lf	Ø	(cacoc ca|ca|c cucuwc)	Ø	Ø
s;fr	Ø	Ø	Ø	Ø
h-dq	Ø	Ø	Ø	Ø
fhm	Ø	Ø	Ø	Ø
ɛlm	Ø	Ø	Ø	Ø
qdm	Ø	Ø	Ø	Ø
svrb	Ø	Ø	Ø	Ø
rkb	Ø	Ø	Ø	Ø
mr:	Ø	Ø	Ø	Ø
sɛd	Ø	Ø	Ø	Ø
rjl	Ø	Ø	Ø	Ø
s;dq	Ø	Ø	Ø	Ø
nfd-	Ø	Ø	Ø	Ø
jbl	Ø	Ø	Ø	Ø
h-mr	Ø	Ø	Ø	Ø
zrq	Ø	Ø	Ø	Ø
:mm	Ø	Ø	Ø	Ø

G. THE P-DATABASE: A DICTIONARY OF PARTICLES AND ADVERBS

a. A Dictionary of Free Prepositions

	HTY	DFN	GVT	CMK	CMK2	CAT	CAT2	MDG	TYP	MODE
fiy	Ø	Ø	obm	Ø	Ø	Pr	Ø	Ø	Ø	Ø
mino	Ø	Ø	obm	Ø	Ø	Pr	Ø	Ø	Ø	Ø
:ilay	Ø	Ø	obm	Ø	Ø	Pr	Ø	Ø	Ø	Ø
:ilayo	Ø	Ø	obm	Ø	Ø	Pr	Ø	Ø	bpr	Ø
ɛano	Ø	Ø	obm	Ø	Ø	Pr	Ø	Ø	Ø	Ø
ɛalay	Ø	Ø	obm	Ø	Ø	Pr	Ø	Ø	Ø	Ø
ɛalayo	Ø	Ø	obm	Ø	Ø	Pr	Ø	Ø	bpr	Ø
maɛa	Ø	Ø	obm	Ø	Ø	Pr	Ø	Ø	Ø	Ø
ɛinoda	Ø	Ø	obm	Ø	Ø	Pr	Ø	Ø	Ø	Ø
laday	Ø	Ø	obm	Ø	Ø	Pr	Ø	Ø	Ø	Ø
ladayo	Ø	Ø	obm	Ø	Ø	Pr	Ø	Ø	bpr	Ø

fawoqa	Ø	Ø	obm	Ø	Ø	Pr	Ø	Ø	Ø	Ø
baɛoda	Ø	Ø	obm	Ø	Ø	Pr	Ø	Ø	Ø	Ø
tah-ota	Ø	Ø	obm	Ø	Ø	Pr	Ø	Ø	Ø	Ø
:ama|ma	Ø	Ø	obm	Ø	Ø	Pr	Ø	Ø	Ø	Ø
qabola	Ø	Ø	obm	Ø	Ø	Pr	Ø	Ø	Ø	Ø
wara|:a	Ø	Ø	obm	Ø	Ø	Pr	Ø	Ø	Ø	Ø

b. A Dictionary for Other (Free) Particles

	HTY	DFN	GVT	CMK	CMK2	CAT	CAT2	MDG	TYP	MODE
lamo	Ø	Ø	Ø	Ø	Ø	JP	Ø	jus	Ø	Ø
lano	Ø	Ø	Ø	Ø	Ø	SP	Ø	sub	Ø	Ø
h-attay	Ø	Ø	Ø	Ø	Ø	SP	Ø	sub	Ø	Ø
kayo	Ø	Ø	Ø	Ø	Ø	SP	Ø	sub	Ø	Ø
:inna	Ø	Ø	acm	Ø	Ø	AP	Ø	Ø	Ø	Ø
:anna	Ø	Ø	acm	Ø	Ø	AP	Ø	Ø	Ø	Ø
la|	Ø	Ø	acm	Ø	Ø	EP	Ø	Ø	Ø	Ø
qado	Ø	Ø	Ø	Ø	Ø	PD	Ø	ind	Ø	Ø
sawofa	Ø	Ø	Ø	Ø	Ø	FD	Ø	ind	Ø	Ø
halo	Ø	Ø	Ø	Ø	Ø	Q2	Ø	Ø	Ø	Ø
ma|	Ø	dfp	Ø	ncp	Ø	JQ	Ø	Ø	Ø	int
mano	Ø	dfp	Ø	ncp	Ø	HQ	Ø	Ø	Ø	int
matay	Ø	dfp	Ø	ncp	Ø	QP	Ø	Ø	Ø	int
:ayona	Ø	dfp	Ø	ncp	Ø	QP	Ø	Ø	Ø	int
kayofa	Ø	dfp	Ø	ncp	Ø	QP	Ø	Ø	Ø	int
ma|d-a|	Ø	dfp	Ø	ncp	Ø	QP	Ø	Ø	Ø	int

c. A Dictionary for Other (Bound) Particles

	HTY	DFN	GVT	CMK	CMK2	CAT	CAT2	MDG	TYP	MODE
:a	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	int
wa	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	int
fa	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	int

d. A Dictionary of Adverbs

	HTY	DFN	GVT	CMK	CMK2	CAT	CAT2	MDG	TYP	MODE
huna|	Ø	dfp	Ø	Ø	Ø	LD	Ø	Ø	Ø	Ø
huna|ka	Ø	dfp	Ø	Ø	Ø	LD	Ø	Ø	Ø	Ø
huna|lika	Ø	dfp	Ø	Ø	Ø	LD	Ø	Ø	Ø	Ø
:alo:a|na	Ø	Ø	Ø	Ø	Ø	AD	Ø	Ø	Ø	Ø
:abada+	Ø	Ø	Ø	Ø	Ø	AD	Ø	Ø	Ø	Ø
baɛodu	Ø	Ø	Ø	Ø	Ø	AD	Ø	Ø	Ø	Ø

==0==<*****>==0==

Appendix C
PROGRAM DESCRIPTION: SCHEMES 7 TO 36

ARGUMENTS: an inputword and a rootlist.
OBJECTIVE:
EXTRACT a preliminary or raw non-vocalic form from the inputword then perform simple transformations on it, verify its MEMBERship in the rootlist and ASSIGN to it the appropriate TRANSITIVITY value.
LOCAL VARIABLES: *a*, *b*, L, LL, yy, rr, qq, and ss.
BASIC METHOD:
† SET ss to the SCRIPTed inputword,
† SET cc to the MKWORD of the INTERSECTION of ss with the consonants,
† SET yy to the MKWORD of the INTERSECTION of ss with the consonants and semivowels,
† SET rr to the SCRIPTed cc,
† SET qq to the SCRIPTed yy,
† SET L to the LENGTH of rr, and
† SET LL to the LENGTH of qq, then
STEP I:
IF L or LL are GREATER than or EQUAL to 6, or IF L or LL are strictly LESS than 2, then FAIL.
STEP II:
IF the 2nd and 3rd Xers of qq are identical and the 2nd Xer is a semivowel, then

1. IF cc is a MEMBER of the rootlist, then

   (a) IF cc is in the Intensive Root Dictionary (IRD), then ASSIGN the TRVY of cc to yy and RETURN yy.

   (b) IF cc is in the Causative Root Dictionary (CRD), then PROMOTE the TRVY of cc and ASSIGN it to yy as follows:
   ⊘ 'kn', "to be", cotransitive →↠ monotransitive,
   ⊘ 's;r', "to become", cotransitive →↠ xtransitive,
   ⊘ 's:', "to sadden", monotransitive →↠ ptransitive2,
   ⊘ intransitive →↠ monotransitive,
   ⊘ monotransitive →↠ ditransitive, and RETURN yy.

   (c)/2. ELSE FAIL.

STEP III:
IF the 3rd and 4th Xers of qq are identical and the 3rd Xer is a semivowel, then

1. IF the initial Xer of qq is <m>, then DO:
   ⊘ SET *a* to MKWORD of qq with the initial <m> DELETed, and
   ⊘ SET *b* to MKWORD of rr with the initial <m> DELETed, then

   (a) DO aa = [IF *b* is a MEMBER of the rootlist, then

      i. IF *b* is in IRD, then ASSIGN the TRVY of *b* to *a* and RETURN *a*.

      ii. IF *b* is in CRD, then PROMOTE the TRVY of *b* and ASSIGN it to *a* as in II.1.(b) and RETURN *a*.

      iii. ELSE FAIL].

   (b)/2. ELSE FAIL.

STEP IV:
IF cc is a MEMBER of the rootlist and L is LESS than or EQUAL to 4 then RETURN cc.
STEP V:
IF either the 1st and 2nd or the 2nd and 3rd Xers of rr are identical, and the 2nd Xer of qq is not a semivowel, then

1. IF the 1st and 2nd Xers of rr are both <m>, then SET *a* to MKWORD of rr with the 1st <m> DELETed, then

   (a) DO bb = [IF *a* is a MEMBER of the rootlist and L is LESS than or EQUAL to 4, then RETURN *a*].

   (b) ELSE FAIL.

2. ELSE SET *a* to MKWORD of rr with the 2nd Xer DELETed, then

   (a) IF *a* is a MEMBER of the rootlist, then

      i. IF *a* is in IRD, then ASSIGN the TRVY of *a* to cc, & RETURN cc.

      ii. IF *a* is in CRD, then PROMOTE the TRVY of *a* and ASSIGN it to cc as in II.1.(b) and RETURN cc.

      iii./(b)/3. ELSE FAIL.

STEP VI:
If the initial Xer of rr is <m>, then

1. IF either the 2nd and 3rd or the 3rd and 4th Xers of rr are identical, then DO:
   ⊘ SET *a* to MKWORD of rr with the initial <m> DELETed, and
   ⊘ SET *b* to MKWORD of rr with the 3rd Xer DELETed, then

   (a) DO aa.

   (b) IF the final Xer of rr is <t>, then DO:
      ⊘ SET *a* to MKWORD of rr with the final <t> DELETed, and
      ⊘ SET *b* to MKWORD of rr with the 1st and 3rd Xers and final <t> DELETed, then

      i. DO aa.

      ii. ELSE FAIL.

   (c) IF *a* is a MEMBER of the rootlist, then RETURN *a*.

2. ELSE SET *a* to MKWORD of rr with the 1st Xer <m> DELETed, then

   (a) DO bb.

   (b) IF the final Xer of rr is <t>, then SET *a* to MKWORD of rr with the final <t> DELETed, then

      i. IF *a* is a MEMBER of the rootlist, then RETURN *a*.

      ii./(c)/3. ELSE FAIL.

STEP VII:
IF the 2nd Xer of rr is <:>, then SET *a* to MKWORD of rr with the 2nd Xer <:> DELETed, then

1. IF *a* is a MEMBER of the rootlist, then RETURN *a*.

2./*STEP VIII.* ELSE FAIL.

SCHEME 7: EXROOT: A Procedural Scheme for Root EXTRACTion

ARGUMENTS:
an inputword x and a Global Pattern List y which is a set of association lists of roots and patterns.
OBJECTIVE:
obtain a valid Verb DERIVation from the inputword x by EXTRACTing a root from it, retrieving the list of patterns ASSOCIATed with the root in the database and then searching that list for one pattern that MATCHes the inputword x. The DERIVation is of the irregular first- and second-PERSON type for DERIVE1, of the irregular third-PERSON type for DERIVE2, and of the regular types for all PERSONS for DERIVE3.
LOCAL VARIABLES: pp, root, and root2.
BASIC METHOD:
STEP 1:
SET root to EXROOT of the inputword from the list of irregular Verb roots: NRVB.
STEP 2:
IF the GPL y is empty, then FAIL.
STEP 3: %%% DERIVE1.
IF root is not NULL, then SET pp to the 2nd element of the LPL obtained by ASSOCIATing root with the GPL y, then
(a) IF pp MATCHes x, then %%% using MATCHX.
DO aa = [RETURN a list which is the name of the LPL, root, and pp].
(b) ELSE SET the GPL y to the REST of y, then GOTO *STEP 2.*
STEP 4: %%% DERIVE2.
IF root is not NULL, then SET pp to the 1st element of the LPL obtained by ASSOCIATing root with the GPL y, then
(a) IF root is not a MEMBER of the Augmented Df.Ho. roots, then
i. IF pp MATCHes x, then DO aa. %%% using MATCHX.
ii. ELSE SET the GPL y to the REST of y, then GOTO *STEP 2.*
STEP 5: %%% DERIVE3.
SET root to EXROOT of x from RRVB.
STEP 6:
SET root2 to EXROOT of x from NRVB.
STEP 7:
IF the GPL y is empty, then FAIL.
STEP 8:
IF root is not NULL, then SET pp to the 1st element of the LPL obtained by ASSOCIATing root with the GPL y, then
(a) IF pp MATCHes x, then DO aa. %%% using MATCHX.
(b) ELSE SET the GPL y to the REST of y, then GOTO *STEP 7.*
STEP 9:
IF root2 is not NULL, then SET pp to the 1st element of the LPL obtained by ASSOCIATing root2 with the GPL y, then
(a) IF root2 is a MEMBER of the Augmented Df.Ho. roots, then
i. IF pp MATCHes x, then %%% using MATCHX.
RETURN a list which is the name of the LPL, root2, and pp.
ii. ELSE SET y to the REST of y, then GOTO *STEP 7.*

SCHEME 8: DERIVE: An Iterative Scheme for Verb DERIVation

ARGUMENTS: an inputword *v* for Verb.
OBJECTIVE:
recognize a Simple Perfect CF by identifying a perfect V-Suffix and conjugating the centre of the inputword *v* in the perfect active or passive form.
LOCAL VARIABLES: *i*, *a*, *b*, *s*, LL, rr, and ans.
BASIC METHOD:
STEP A:
IF the final Xer of *v* is in the set of perfect suffix boundaries as specified, then DO:
%%% entry condition.
STEP I:
SET *i* to 1, then
STEP II:
IF *i* is GREATER than or EQUAL to 7, then FAIL.
STEP III:
IF *i* is strictly LESS than 7, then SET *s* to SEGMV of *v* using the index *i* and SET *a* to the 1st and *b* the 2nd elements of *s*, then %%% *a* is the suffix, *b* the centre.

1. SET ans to be DERIVE1 of *b* from the GPL: "PATLISTP", and IF ans is not NULL, then SET LL to the 1st and rr the 2nd elements of ans, then %%% ans is a list: LL: the LPL, rr: the root, and the pattern.

   (a) IF rr is in the Defective root set: {svq, gn}, then

      i. IF *a* is EQUAL to 'uw|', then DO aa = [PASSON the NGP of *a*, the CAT & TRVY of rr & the VOC of LL to *v*, ASSIGN to *v* the TNS: prf, & RETURN a list of *b*, *a*, & ans].

      ii. ELSE FAIL.

   (b) IF rr is the Defective root 'gnn', then

      i. IF *a* is EQUAL to 'uw|', then DO aa.

      ii. IF LL has the VOICE value: active, then

         IF *a* is EQUAL to 'ay', then %%% singular Df. suf.
      DO bb = [ASSIGN to *v* the NGP, TNS, & VOC: 3 (M) S, prf, act, PASSON the CAT & TRVY of rr, & RETURN a list of *b*, *a*, & ans].

         IF *a* is in the 3rd PERSON suffixes: {ato, ata|}, then
      DO cc = [PASSON the NGP of *a*, the CAT & TRVY of rr to *v*, ASSIGN to *v* the TNS & VOC: prf, act, & RETURN a list of *b*, *a*, & ans].

         /iii. ELSE FAIL.

   (c) IF rr is in the Defective root set: {bn, t-n, d-k, m:, bnn, t-nn, d-kk}, then

      i. IF LL has the VOICE value: passive, then

         IF *a* is EQUAL to 'uw|', then DO dd = [PASSON the NGP of *a*, the CAT & TRVY of rr to *v*, ASSIGN to *v* the TNS & VOC: prf, pas, & RETURN a list

of *b*, *a*, & ans].

ELSE FAIL.

ii. IF LL has the VOICE value: active, then

IF *a* is EQUAL to 'ay', then DO bb.

IF *a* is in the 3rd PERSON suffixes: {ato, ata|}, then DO cc.

IF *a* is EQUAL to 'awo|', then %%% plural Df. suf.
DO ee = [ASSIGN to *v* the NGP, TNS, & VOC: 3 (M) P, prf, act, PASSON the CAT & TRVY of rr, & RETURN a list of *b*, *a*, & ans].

/iii. ELSE FAIL.

(d) IF rr is in the Defective root set: {:b, :x, ns}, then

i. IF LL has the VOICE value: passive, then

IF *a* is EQUAL to 'uw|', then DO dd.

ELSE FAIL.

ii. IF LL the VOICE value: active, then

IF *a* is EQUAL to 'a|', then DO bb.

IF *a* is in the 3rd PERSON suffixes: {ato, ata|}, then DO cc.

IF *a* is EQUAL to 'awo|', then DO ee.

/iii. ELSE FAIL.

(e) IF *a* is EQUAL to 'a|', then %%% Disambiguation of PRS value for 'a|'.
DO ff = [PASSON the NG of *a*, the CAT & TRVY of rr & the VOC of LL to *v*, ASSIGN to *v* the PRS & TNS: 3rd, prf, & RETURN a list of *b*, *a*, & ans].

(f) IF *a* is in the set of other 3rd PERSON irregular Defective suffixes, then DO aa.

(g) ELSE FAIL.

2. SET ans to be DERIVE2 of *b* from the GPL: "PATLISTP", and IF ans is not NULL, then SET LL to the 1st and rr the 2nd elements of ans, then

(a) IF rr is in the Defective root set: {svq, gn, gnn}, then

i. IF *a* is not EQUAL to 'uw|', then %%% 'uw|' is excluded here.

IF *a* is EQUAL to 'a|', then DO ff.

IF *a* is in the set of regular non-Defective suffixes for all PERSONS, then DO aa.

/ii. ELSE FAIL.

(b) IF rr is in the Defective root set: {bn, t-n, d-k, m:, :b, :x, ns, bnn, t-nn, d-kk}, then

i. IF LL has the VOICE value: passive, then

IF *a* is not EQUAL to 'uw|', then %%% 'uw|' is excluded here.

IF *a* is EQUAL to 'a|', then ASSIGN to *v* the VOC: pas, & DO gg = [PASSON the NG of *a*, the CAT & TRVY of rr to *v*, ASSIGN to *v* the PRS & TNS: 3rd, prf, & RETURN a list of *b*, *a*, & ans].

IF *a* is in the set of regular non-Defective suffixes for all PERSONS, then ASSIGN to *v* the VOC: pas, & DO hh = [PASSON the NGP of *a*, the CAT & TRVY of rr to *v*, ASSIGN to *v* the TNS: prf, & RETURN a list of *b*, *a*, & ans].

ELSE FAIL.

ii. IF LL has the VOICE value: active, then

IF *a* is EQUAL to 'a|', then ASSIGN to *v* the VOC: act, & DO gg.

IF *a* is in the set of 1st and 2nd PERSON irregular Df. suffixes, then ASSIGN to *v* the VOC: act, & DO hh.

/iii. ELSE FAIL.

(c) IF *a* is in the set of all irregular Defective suffixes except 'a|', then DO aa.

(d) IF rr is in the irregular Sound root set: {ɛmmm, jddd}, then

i. IF *a* is EQUAL to 'a|', then DO ff.

ii. IF *a* is in the set of other 3rd PERSON irregular Defective suffixes, then DO aa.

iii./(e) ELSE FAIL.

3. SET ans to be DERIVE3 of *b* from the GPLs: "PATLISTP", and IF ans is not NULL, then SET LL to the 1st and rr the 2nd elements of ans, then

(a) IF *a* is EQUAL to 'a|', then DO ff.

(b) IF *a* is in the set of regular non-Defective suffixes for all PERSONS, then DO aa.

(c) ELSE SET *i* to *i* PLUS 1 and GOTO *STEP II.* %%% increment index by 1.

STEP IV./STEP B. ELSE FAIL.

SCHEME 9: MKPERFECT: An Iterative Left-to-Right Scheme for Perfect CF Recognition

---

ARGUMENTS: an inputword *v* for Verb.
OBJECTIVE:
recognize a Simple Imperfect CF by identifying an imperfect prefix, an imperfect suffix and conjugating the centre of the inputword *v* in the imperfect active or passive form.
LOCAL VARIABLES: *i*, *a*, *b*, *c*, *s*, LL, rr, and ans.
BASIC METHOD:
STEP A:
IF the initial Xer of *v* is in the set of imperfect prefix boundaries and the final Xer of *v* is in the set of imperfect suffix boundaries as specified, then DO: %%% entry condition.
STEP I:
SET *i* to 1, then
STEP II:
IF *i* is GREATER than or EQUAL to 6, then FAIL.
STEP III:
IF *i* is strictly LESS than 6, then SET *s* to MNSEGM 1 & *i* of *v* and SET *a* to the 1st, *b* the 2nd, & *c* the 3rd elements of *s*, then
%%% *a* is the prefix, *b* the suffix, & *c* the main segment or centre.

1. SET ans to be DERIVE1 of *c* from the GPL: "PATLISTI", and IF ans is not NULL, then SET LL to the 1st and rr the 2nd elements of ans, then %%% ans is a list: LL: the LPL, rr: the root, and the pattern.

   (a) IF rr is in the Defective root set: {:b, :x, ns}, then

      i. IF LL has the VOICE value: passive, then

         IF *b* is EQUAL to <a>, then

            IF *a* is in the imperfect prefixes: {:, t, y, n}, then
         ASSIGN to *v* the MMK: jus, & DO aa = [PASSON the NGP of *a*, the CAT & TRVY of rr to *v*, ASSIGN to *v* the TNS & VOC: imp, pas, & RETURN a list of *a*, *c*, *b*, & ans].

            ELSE FAIL.

         IF *b* is EQUAL to 'ay', then

            IF *a* is in the imperfect prefixes: {:, t, y, n}, then ASSIGN to *v* the MMK: ids, & DO aa.

            ELSE FAIL.

         IF *a* is EQUAL to <t> and *b* is in the Defective suffixes: {awo|, awona}, then ASSIGN to *v* the VOC: pas, & DO bb = [PASSON the NGP & MMK of *a*, the CAT & TRVY of rr to *v*, ASSIGN to *v* the TNS: imp, & RETURN a list

of *a*, *c*, *b*, & ans].

IF *a* is EQUAL to <y> and *b* is in the Defective suffixes: {awo|, awona}, then ASSIGN to *v* the VOC: pas, & DO cc = [PASSON the NBR & MMK of *b*, the GDR & PRS of *a*, the CAT & TRVY of rr to *v*, ASSIGN to *v* the TNS: imp, & RETURN a list of *a*, *c*, *b*, & ans].

ELSE FAIL.

ii. IF LL has the VOICE value: active, then

IF *b* is EQUAL to <u>, then

IF *a* is in the imperfect prefs: {:, t, y, n}, then ASSIGN to *v* the MMK: jus, & DO dd = [PASSON the NGP of *a*, the CAT & TRVY of rr to *v*, ASSIGN to *v* the TNS & VOC: imp, act, & RETURN a list of *a*, *c*, *b*, & ans].

ELSE FAIL.

IF *b* is EQUAL to 'uw', then

IF *a* is in the imperfect prefix set: {:, t, y, n}, then ASSIGN to *v* the MMK: ind, & DO dd.

ELSE FAIL.

IF *a* is EQUAL to <t> and *b* is in the suffix set: {iy, iyna}, then ASSIGN to *v* the VOC: pas, & DO ee = [PASSON the NBR & MMK of *b*, the CAT & TRVY of rr to *v*, ASSIGN to *v* the GDR, PRS, & TNS: 2 (F), imp, & RETURN a list of *a*, *c*, *b*, & ans].

/iii. ELSE FAIL.

(b) IF rr is in the Defective root set: {gn, m:, svq, d-k}, then

i. IF *b* is EQUAL to <a>, then

IF *a* is in the imperfect prefix set: {:, t, y, n}, then ASSIGN to *v* the MMK: jus, & DO ff = [PASSON the NGP of *a*, the CAT & TRVY of rr, the VOC of LL to *v*, ASSIGN to *v* the TNS: imp, & RETURN a list of *a*, *c*, *b*, & ans].

ELSE FAIL.

ii. IF *b* is EQUAL to 'ay', then

IF *a* is in the imperfect prefix set: {:, t, y, n}, then ASSIGN to *v* the MMK: ids, & DO ff.

ELSE FAIL.

iii. IF *a* is EQUAL to <t> and *b* is in the Defective suffixes: {awo|, awona}, then PASSON to *v* the VOC of LL, & DO bb.

iv. IF *a* is EQUAL to <y> and *b* is in the Defective suffixes: {awo|, awona}, then PASSON to *v* the VOC of LL, & DO cc.

v. ELSE FAIL.

(c) IF rr is in the Defective root set: {bnn, t-nn, gnn}, then

i. IF *b* is in the suffix set: {a, i}, then

IF *a* is in the imperfect prefix set: {:, t, y, n}, then PASSON the NGP of *a*, the CAT & TRVY of rr, the VOC of *b* to *v*, ASSIGN to *v* the TNS & MMK: imp, jus, & RETURN a list of *a*, *c*, *b*, & ans.

ELSE FAIL.

ii. IF *b* is EQUAL to 'ay', then

IF *a* is in the imperfect prefix set: {:, t, y, n}, then PASSON to *v* the MMK of *b*, & DO aa.

ELSE FAIL.

iii. IF *b* is EQUAL to 'iy', then

IF *a* is in the imperfect prefix set: {:, y, n}, then ASSIGN to *v* the MMK: ind, & DO dd.

IF *a* is EQUAL to <t>, then DO gg = [PASSON the CAT & TRVY of rr to *v*, ASSIGN to *v* the NGP, TNS, MMK, & VOC: XR (MF) S, imp, xds, act, & RETURN a list of *a*, *c*, *b*, & ans].

ELSE FAIL.

iv. IF *a* is EQUAL to <t> and *b* is in the set: {uw|, awo|, uwna, awona}, then PASSON the MMK & VOC of *b*, the CAT & TRVY of rr to *v*, ASSIGN to *v* the NGP & TNS: 2 (M) P, imp, & RETURN a list of *a*, *c*, *b*, & ans.

v. IF *a* is EQUAL to <y> and *b* is in the set: {uw|, awo|, uwna, awona}, then PASSON the NBR, MMK, & VOC of *b*, the GDR & PRS of *a*, the CAT & TRVY of rr to *v*, ASSIGN to *v* the TNS: imp, & RETURN a list of *a*, *c*, *b*, & ans.

vi. ELSE FAIL.

(d) IF rr is in the Defective root set: {bn, t-n, d-k}, then

i. IF LL has the VOICE value: passive, then

IF *b* is EQUAL to <a>, then

IF *a* is in the imperfect prefix set: {:, t, y, n}, then ASSIGN to *v* the MMK: jus, & DO aa.

ELSE FAIL.

IF *b* is EQUAL to 'ay', then

IF *a* is in the imperfect prefix set: {:, t, y, n}, then ASSIGN to *v* the MMK: ids, & DO aa.

ELSE FAIL.

IF *a* is EQUAL to <t> and *b* is in the Df. suffixes: {awo|, awona}, then ASSIGN to *v* the VOC: pas, & DO bb.

IF *a* is EQUAL to <y> and *b* is in the Df. suffixes: {awo|, awona}, then ASSIGN to *v* the VOC: pas, & DO cc.

ELSE FAIL.

ii. IF LL has the VOICE value: active, then

IF *b* is EQUAL to <i>, then

IF *a* is in the imperfect prefix set: {:, t, y, n}, then ASSIGN to *v* the MMK: jus, & DO dd.

ELSE FAIL.

IF *b* is EQUAL to 'iy', then

IF *a* is in the imperfect prefix set: {:, y, n}, then ASSIGN to *v* the MMK: ind, & DO dd.

IF *a* is EQUAL to <t>, then DO gg.

ELSE FAIL.

IF *a* is EQUAL to <t> and *b* is in the set: {uw|, uwna}, then PASSON the MMK of *b* & the CAT & TRVY of rr to *v*, ASSIGN to *v* the NGP, TNS, & VOC: 2 (M) P, imp, act, & RETURN a list of *a*, *c*, *b*, & ans.

IF *a* is EQUAL to <y> and *b* is in the set: {uw|, uwna}, then ASSIGN to *v* the VOC: act, & DO cc.

/iii. ELSE FAIL.

(e) IF *a* is EQUAL to <y> and *b* is in the set: {ona, na}, then PASSON the VOC of LL to *v*, & DO bb.

(f) IF *a* is EQUAL to <t> and *b* is in the set: {ona, na}, then PASSON the VOC of LL to *v*, & DO hh = [PASSON the CAT & TRVY of **rr**, the MMK of *b*, ASSIGN to *v* the NGP & TNS: 2 (F) P, imp, & RETURN a list of *a*, *c*, *b*, & ans].

(g) IF *b* is EQUAL to <o>, then

i. IF *a* is in the imperfect prefix set: {:, t, y, n}, then PASSON to *v* MMK of *b*, & DO ff.

ii./(h) ELSE FAIL.

2. SET ans to be DERIVE2 of *c* from the GPL: "PATLISTI", and IF ans is not NULL, then SET LL to the 1st and rr the 2nd elements of ans, then

(a) IF rr is in the Defective root set: {gn, m:, svq, d-k}, then

i. IF *a* is EQUAL to <t> and *b* is EQUAL to <o>, then PASSON the VOC of LL to *v*, & DO ii = [PASSON the CAT & TRVY of rr, ASSIGN to *v* the NGP, TNS, & MMK: 2 (F) S, imp, sjj, & RETURN a list of *a*, *c*, *b*, & ans].

ii. IF *a* is EQUAL to <t> and *b* is in the set: {ona, na}, then PASSON the VOC of LL to *v*, & DO jj = [PASSON the CAT & TRVY of rr, the MMK of *b*, ASSIGN to *v* the NGP & TNS: 2 (F) SP, imp, & RETURN a list of *a*, *c*, *b*, & ans].

iii. IF either *a* is EQUAL to <t> and *b* is in the set: {a|, a|ni}, or *a* is EQUAL to <y> and *b* is in the set: {ona, na}, then PASSON the VOC of LL to *v*, & DO bb.

iv. IF *a* is EQUAL to <y> and *b* is in the set: {a|, a|ni}, then PASSON the VOC of LL to *v*, & DO cc.

v. ELSE FAIL.

(b) IF rr is in the Defective root set: {bn, t-n, :b, :x, ns, bnn, t-nn, gnn, d-kk}, then

i. IF *a* is EQUAL to <t> and *b* is EQUAL to <o>, then

IF LL has the VOICE value: passive, then ASSIGN to *v* the VOC: pas, & DO ii.

ELSE FAIL.

ii. IF *a* is EQUAL to <t> and *b* is in the set: {ona, na}, then

IF LL has the VOICE value: passive, then ASSIGN to *v* the VOC: pas, & DO jj.

IF LL has the VOICE value: active, then ASSIGN to *v* the VOC: act, & DO hh.

ELSE FAIL.

iii. IF *b* is EQUAL to <a>, then

IF *a* is in the imperfect prefix set: {:, t, y, n}, then

IF LL has the VOICE value: active, then ASSIGN to *v* the MMK: sub, & DO dd.

ELSE FAIL.

iv. IF either *a* is EQUAL to <t> and *b* is in the set: {a|, a|ni}, or *a* is EQUAL to <y> and *b* is in the set: {ona, na}, then PASSON the VOC of LL to *v*, & DO bb.

v. IF *a* is EQUAL to <y> and *b* is in the set: {a|, a|ni}, then PASSON the VOC of LL to *v*, & DO cc.

vi. ELSE FAIL.

(c) IF *b* is in the suffix set: {u, a}, then

i. IF *a* is in the imperfect prefix set: {:, t, y, n}, then PASSON the MMK of *b* to *v*, & DO ff.

ii. ELSE FAIL.

(d) IF *a* is EQUAL to <t> and *b* is in the set: {iy, iyna}, then PASSON the VOC of LL to *v*, & DO ee.

(e) IF either *a* is EQUAL to <t> and *b* is in the set: {a|, a|ni}, or *a* is EQUAL to <t> and *b* is in the set: {uw|, uwna}, then PASSON the NBR, GDR, & MMK of *b*, the VOC of LL & the CAT & TRVY of rr to *v*, ASSIGN to *v* the PRS: 2nd & the TNS: imp, & RETURN a list of *a*, *c*, *b*, & ans.

(f) IF either *a* is EQUAL to <y> and *b* is in the set: {a|, a|ni}, or *a* is EQUAL to <y> and *b* is in the set: {uw|, uwna}, then PASSON the VOC of LL to *v*, & DO cc.

(g) ELSE FAIL.

3. SET ans to be DERIVE3 of *c* from the GPL: "PATLISTI", and IF ans is not NULL, then SET LL to the 1st and rr the 2nd elements of ans, then

(a) IF *b* is in the suffix set: {u, a, o}, then

i. IF *a* is in the imperfect prefix set: {:, t, y, n}, then PASSON the MMK of *b* to *v*, & DO ff.

ii. ELSE FAIL.

(b) IF *a* is EQUAL to <t> and *b* is in the set: {iy, iyna}, then PASSON the VOC of LL to *v*, & DO ee.

(c) IF either *a* is EQUAL to <t> and *b* is in the set: {uw|, uwna}, then PASSON the NBR, GDR, & MMK of *b*, the VOC of LL & the CAT & TRVY of rr to *v*, ASSIGN to *v* the PRS: 2nd & the TNS: imp, & RETURN a list of *a*, *c*, *b*, & ans.

(d) IF either *a* is EQUAL to <t> and *b* is in the set: {a|, a|ni}, or *a* is EQUAL to <y> and *b* is in the set: {ona, na}, then PASSON the VOC of LL to *v*, & DO bb.

(e) IF *a* is EQUAL to <t> and *b* is in the set: {ona, na}, then PASSON the VOC of LL to *v*, & DO hh.

(f) IF either *a* is EQUAL to <y> and *b* is in the set: {a|, a|ni}, or *a* is EQUAL to <y> and *b* is in the set: {uw|, uwna}, then PASSON the VOC of LL to *v*, & DO cc.

(g) ELSE FAIL.

4. SET *i* to *i* PLUS 1 and GOTO *STEP II.* %%% increment index by 1.

STEP IV./STEP B. ELSE FAIL.

SCHEME 10: MKIMPERFECT: An Iterative Left-to-Right-to-Left Scheme for Imperfect CF Recognition

---

ARGUMENTS: an inputword *v* for Verb.
OBJECTIVE:
† supervise and enforce the observation of the Graphotactic Conditions governing the morphological sequences: c_1c_1 and c_1oc_2 and FILTER out sequences of the type: $*c_1oc_1$ and $*c_1c_2$ in a Perfect CF (FILTER1) or in an Imperfect CF (FILTER2).
† resolve Simple CF homonymy for Imperfect CFs (FILTER2).
LOCAL VARIABLES: *a*, *b*, *c*, *d*, *e*, *f*, *g*, *s*, ss, and rr.
BASIC METHOD: %%% FILTER1.
STEP A:
SET *s* to MKPERFECT of *v* and IF *s* is not NULL, SET *a* to the 2nd element of *s*, then %%% *a* is the perfect V-Suffix.
STEP I:
IF the 3rd element of *s* is in the root set: {bt, kn, d/nn, h-sn, t-mn, skn, byyt, kwwn, h-ssn, t-mmn, skkn}, then

1. IF either *a* is in the set: {ona, na} or is 1st or 2nd PERSON, then

   (a) IF the initial Xer of *a* is in the set: {n, t}, then

      i. IF the initial Xer of *a* is identical with the final Xer of the centre, then RETURN *s*.

      ii. ELSE DO aa = [embed *v* in the error message: "error1", and RETURN "error1"].

   (b) IF the initial Xer of *a* is <o>, then

      i. IF the 2nd Xer of *a* is not identical with the final Xer of the centre, then RETURN *s*.

      ii. ELSE DO bb = [embed *v* in the error message: "error2", and RETURN "error2"].

      iii. ELSE FAIL.

2. ELSE RETURN *s*.

STEP II:
IF the initial Xer of *a* is in the set: {n, t}, then DO aa.
STEP III: ELSE RETURN *s*.
STEP B. ELSE FAIL.
BASIC METHOD: %%% FILTER2.
STEP A:
SET *s* to MKIMPERFECT of *v* and IF *s* is not NULL, SET *b* to the 3rd, *c* the 2nd & rr the 4th elements of *s*, then %%% *b* is the imperfect V-Suffix, *c* the V-Centre & rr the root.
STEP I:
IF rr is in the root set: {kn, d/nn, h-sn, t-mn, skn, kwwn, h-ssn, t-mmn, skkn}, then

1. IF *b* is in the set: {ona, na}, then

   (a) IF the initial Xer of *b* is <n>, then

      i. IF the initial Xer of *b* is identical with the final Xer of *c*, then RETURN *s*.

      ii. IF the initial Xer of *b* is <o>, then

         IF the 2nd Xer of *b* is not identical with the final Xer of *c*, then RETURN *s*.

         ELSE DO bb.

         /iii./(b) ELSE FAIL.

2. ELSE RETURN *s*.

STEP II:
IF the initial Xer of *b* is not <n>, then RETURN *s*.
STEP III:
IF the initial Xer of *b* is <n>, then

1. IF rr is in the root set: {:b, :x, ns, bn, t-n, bnn, t-nn, gnn, d-kk}, then SET ss to SEGMV 2 of *c*, SET *a* to the 1st element of *s*, *d* the 1st & *e* the 2nd elements of ss, then %%% *a* is the imperfect V-Prefix, *d* the final 2 Xers of *c*, & *e* the REST of *c*.

   (a) IF *a* is EQUAL to <t>, then

      i. IF *d* is EQUAL to 'uw', then DO cc = [SET *g* to the 2nd element of SEGMV 2 of the final element of *s*], then
      %%% *b* is 'na' & the final element of *s* is the pattern.
      PASSON the NGP & MMK of 'uwna' to *v* & RETURN a list of *a*, *e*, 'uwna', the LPL, rr, & *g*.

      ii. IF *d* is EQUAL to 'iy', then DO cc, then ASSIGN to *v* the NGP & MMK: 2 (F) S, ind, & RETURN a list of *a*, *e*, 'iyna', the LPL, rr, & *g*.

iii. ELSE FAIL.

(b) IF *a* is EQUAL to <y> and *d* is EQUAL to 'uw', then DO cc, then
SET *f* to MKWORD of *d* & *b*, then %%% *b* is 'na' & *f* is now the suffix.
PASSON the NG & MMK of *f* to *v*, ASSIGN to *v* the PRS: 3rd, & RETURN a list of *a*, *e*, *f*, the LPL, rr, & *g*.

(c) ELSE FAIL.

2. ELSE DO aa.

3./*STEP IV./STEP B.* ELSE FAIL.

SCHEME 11: FILTER1 and 2: The Procedural Schemes for Filtering out Invalid Graphotactic Sequences in Perfect and Imperfect CFs

ARGUMENTS: an inputword *v* for Verb.
OBJECTIVE:
MODIFY the CATEGORIAL and TRANSITIVITY values ASSIGNed by VPARSE taking into account CATEGORIAL HOMONYMY and the PASSIVE VOICE.
LOCAL VARIABLES: rr, pp, and ans.
BASIC METHOD:
STEP A:
SET ans to VPARSE of *v*, then
STEP I:
IF ans is a string, then RETURN ans. %%% a string implies an error.
STEP II:
IF ans is not NULL, then SET rr to the 5th & pp the 6th elements of ans, then %%% rr is the root & pp the pattern.

1. IF *v* has the VOICE value: active, then

   (a) IF rr is the root: 'h-sb', then

      i. IF the penultimate Xer of pp is <i>, then ASSIGN to *v* the TRVY: xtr, & RETURN the 1st three elements of ans.
      %%% These are the Verb and its affixes; the root, pattern, & LPL are no longer needed.

      ii. IF pp is 'cacac', then

         IF *v* has the NGP: 3 (M) S, then MODIFY the CAT of *v* to VN, & DO aa = [ASSIGN to *v* the following Nominal CMK, FLXN & DFN: acm, svm, ndf, & DO bb = [RETURN the 1st three elements of ans]].

         /iii. ELSE DO bb.

   (b) IF rr is the root: 't-mn', then

      i. IF pp is 'cacac', then

         IF *v* has the NGP: 3 (M) S, then MODIFY the CAT of *v* to VN, & DO aa.

         IF the penultimate Xer of pp is <i>, then ASSIGN to *v* the TRVY: mtr, & DO bb.

         ELSE DO bb.

      ii. ELSE FAIL.

   (c) IF rr is the root: 'xd;r' and pp is 'acocac', then

      i. IF *v* has the NBR & GDR values: 1st, sn, then

         IF *v* has the MOOD value: indicative, then ASSIGN to *v* these Nominal CMK, FLXN, & DFN: nmm, svm, ndf, MODIFY the CAT of *v* to VA, & DO bb.

IF v has the MOOD value: indicative-subjunctive, then MODIFY the CAT of v to VA, & DO aa.

ELSE DO bb.

ii. ELSE FAIL.

(d) IF v has the NGP values: 3 (M) S, then

i. IF rr is the root: 'ɛsvr' and pp is 'cacac', then ASSIGN to v the NUMERICAL VALUE of rr, MODIFY the CAT of v to VM, & DO aa.

ii. IF either rr is the root: 'ld' & pp is 'wacac', or rr is in the root set: {d-hb, jbl, qlm} & pp is 'cacac', or rr is in the root set: {ɛmm, jdd} & pp is 'cacc', or rr is in the root set: {xl, bb} & pp is 'ca|c', then MODIFY the CAT of v to VN, & DO aa.

iii./(e) ELSE DO bb.

2. IF v has the VOICE value: passive, then

(a) IF rr is the root: 'h-sb', then ASSIGN to v the TRVY: nctr, and DO bb.

(b) IF rr is in the roots: {d-hb, qdm, nzl}, then

i. IF V has the NGP values: 3 (M) S, then DEMOTE the TRVY of rr & ASSIGN it to v, & DO bb.
%%% TRVY values are DEMOTed as follows:
%%% IF rr is monotransitive $\rightarrow\!\!\!\rightarrow$ intransitive,
%%% IF rr is ditransitive $\rightarrow\!\!\!\rightarrow$ monotransitive,
%%% IF rr is xtransitive $\rightarrow\!\!\!\rightarrow$ cotransitive,
%%% IF rr is ptransitive1 $\rightarrow\!\!\!\rightarrow$ ptransitive,
%%% IF rr is ptransitive2 $\rightarrow\!\!\!\rightarrow$ ptransitive1.

ii. ELSE FAIL.

(c) ELSE DEMOTE the TRVY of rr & ASSIGN it to v, & DO bb.

(d) ELSE DO bb.

3./*STEP III./STEP B.* ELSE FAIL.

SCHEME 12: TVACT1: A Procedural Scheme for Adjusting Categorial and TRANSITIVITY Values

ARGUMENTS: an inputword x for Complex CF.
OBJECTIVES:
† recognize a Complex Perfect or Imperfect CF which can be realized either as a Referential CF—which is itself a Simple CF attached with an Accusative Pronoun—or as a Simple CF without clitic Pronoun.
† perform CATEGORIAL disambiguation and enforce Graphotactic Conditions of affixation by forcing simple transformations.
LOCAL VARIABLES: i, a, b, c, d, s, res, and ans.
BASIC METHOD:
STEP A:
SET res to TVACT1 of x, and IF res is not NULL, then
STEP I:
IF res is a string, then RETURN res.
STEP II: ELSE RETURN x.
STEP B:
If the final Xer of x is in the set: {a, i, o, u, y, |}, then SET i to 2, then
STEP I:
IF i is GREATER than or EQUAL to 6, then FAIL.
STEP II:
IF i is strictly LESS than 6, then SET s to SEGMV i of x, SET a to the 1st & b the 2nd elements of s, then %%% b is the CF & a the Accusative Pronoun.

1. IF a is not in: {iy, ya}, then

   (a) SET ans to TVACT1 of b and IF ans is not NULL, then

      i. IF ans is a string, then RETURN ans.

      ii. IF b has the CATEGORY: VN, then

         IF b is intransitive or ptransitive1, then

         IF a is not 'niy', then DO aa = [PASSON2 the NGP & CMK of b to x, ASSIGN to x the DFN: dfp, and the REF value of a, MODIFY the CAT of x to AN & RETURN a list of b & a].

         ELSE FAIL.

         IF b is monotransitive, then

         IF a is 'niy', then DO bb = [PASSON2 the NGP, TNS, VOC, TRVY, & MMK of b to x, ASSIGN to a the CMK2 & CAT2: acm, CP, ASSIGN to x the REF of a, MODIFY the CAT of x to VR & RETURN a list of b & a].

         ELSE PASSON2 the NGP, CMK, TNS, VOC, TRVY, & MMK of b to x, ASSIGN to a the CMK2 & CAT2: cpm, XP, ASSIGN to x the DFN: dfp, and the REF of a, MODIFY the CAT of x to WV & RETURN a list of b & a.

         ELSE FAIL.

      iii. IF b has the CATEGORY: VM, & DO bb.

iv. IF *b* has the CATEGORY: VA, ASSIGN to *a* the CMK2 & CAT2: obm, GP, & DO aa.

v. IF the final Xer of *b* is <|>, then

IF *b* has the NBR & GDR values: (M) P, then

IF *b* does not have the PRS: 3rd, then DO cc [PASSON2 the property heritage: VPROPLIST of *b* to *x*, ASSIGN to *a* the CMK2 & CAT2: acm, CP, ASSIGN to *x* the REF of *a*, MODIFY the CAT of *x* to VR & RETURN a list of *b* & *a*] .

ELSE FAIL.

ELSE DO cc.

ELSE FAIL.

vi. IF the final Xer of *b* is <o>, then

IF *b* has the NBR & GDR values: (M) P, then

IF *b* does not have the PRS value: 2nd, then DO cc.

ELSE FAIL.

ELSE DO cc.

ELSE FAIL.

vii. ELSE SET *c* to MKWORD of *b* and <|>, then SET ans to TVACT1 of *c* and IF ans is not NULL, then

IF ans is an error RETURN ans.

ELSE DO dd = [PASSON2 the property heritage: VPROPLIST of *c* to *x*, ASSIGN to *a* the CMK2 & CAT2: acm, CP, ASSIGN to *x* the REF of *a*, MODIFY the CAT of *x* to VR & RETURN a list of *b* & *a*].

ELSE FAIL.

viii. IF the 1st element of SEGMV 2 of *b* is EQUAL to 'uw', then SET *c* to MKWORD of *b* after DELETing <w> and then SUBSTITUTing <o> for <u>, then SET ans to TVACT1 of *c* and IF ans is not NULL, then

IF ans is an error RETURN ans.

ELSE DO dd.

/ix./(b) ELSE FAIL.

2. ELSE SET *i* to *i* PLUS 1, then GOTO *STEP I.* %%% increment *i* by 1.

STEP III./STEP C. ELSE FAIL.

SCHEME 13: XPARSE: An Iterative Right-to-Left Scheme for Complex CF Recognition

---

ARGUMENTS: an inputword x for Complex CF.
OBJECTIVE:
monitor the output of XPARSE and enforce the observation of the Vocalic Compatibility and TRANSITIVITY conditions of CF affixation with CPs.
LOCAL VARIABLES: a and ans.
BASIC METHOD:
STEP A:
SET ans to XPARSE of x and IF ans is not NULL, then
STEP I:
IF ans is an error RETURN ans.
STEP II:
IF x has a REFERENCE value NIL, then RETURN ans. %%% x is a Simple CF with no CP attached to it.
STEP III:
ELSE SET a to the 2nd element of ans, then %%% a is the Pronoun.

1. IF x is either Annexed, monotransitive, ditransitive, xtransitive, or ptransitive2, then %%% x is Annexed if it has the CATEGORY: AN.

   (a) IF x is collocutive, then RETURN ans.

   (b) IF x is exlocutive, then

      i. IF a is feminine singular, then RETURN ans.

      ii. IF the final Xer of the 1st element of ans is in the set: {i, y}, then %%% this element is the Simple CF.

         IF the 2nd Xer of a is <i>, then RETURN ans.

         ELSE embed x in the error message: "error3", and RETURN "error3".

         ELSE FAIL.

      iii. IF the final Xer of the 1st element of ans is <o>, then

         IF the penultimate Xer of this 1st element is <y>, then

            IF the 2nd Xer of a is <i>, then RETURN ans.

            ELSE FAIL.

         IF the 2nd Xer of a is <u>, then RETURN ans.

ELSE embed x in the error message: "error9", and RETURN "error9".

ELSE FAIL.

iv. IF the final Xer of the 1st element of ans is in the set: {a, u, w, |}, then

IF the 2nd Xer of a is <u>, then RETURN ans.

ELSE embed x in the error message: "error4", and RETURN "error4".

v./(c)/2./*STEP IV.* ELSE RETURN x embedded in "error6".

STEP B. ELSE FAIL.

SCHEME 14: FILTER3: A Procedural Scheme for Graphotactic and TRANSITIVITY Filtering for CCFs

---

ARGUMENTS: an inputword x for Complex CF.
OBJECTIVES:
† recognize a Perfect or Imperfect Complex CF which is a composite structure of Future Particle (F) + Simple CF + (CP).
† enforce Future Particle MOOD GOVERNMENT conditions.
† perform MOOD and NUMBER disambiguation.
LOCAL VARIABLES: a, b, s, res, and ans.
BASIC METHOD:
STEP A:
SET res to FILTER3 of x, and IF res is not NULL, then RETURN res.
STEP B:
ELSE IF the 1st Xer of x is in the set: {s, l}, then %%% entry condition.
STEP I:
SET s to SEGMV 2 of x and SET a to the 1st & b the 2nd elements of s, then
%%% a is the prefix & b the main CF.

1. IF a is EQUAL to 'sa', then

(a) SET ans to FILTER3 of b, and IF ans is not NULL, then

i. IF ans is a string, then RETURN ans.

ii. IF b has a REFERENCE value NIL and b has a TNS value: imperfect, then

IF b has a MOOD value: indicative or homonymic with the indicative, then

IF b has the CATEGORY VA, then ASSIGN to x the CAT: PVB, & DO aa = [PASSON2 from b to x the property heritage: NGP, TRVY, & VOC, ASSIGN to x the TNS & MMK: imp, ind, & RETURN x].

ELSE ASSIGN to x a CAT which is P attached to the CAT of b, & DO aa.

ELSE FAIL.

iii. IF *b* has the TNS: imperfect and *b* is not Annexed, then

IF *b* has a MOOD value: indicative or homonymic with the indicative, then ASSIGN to *x* the MMK: ind, & DO bb = [PASSON2 from *b* to *x* the property heritage: NGP, TRVY, VOC, DFN, CMK, & REF, ASSIGN to *x* the TNS: imp, and a CAT which is P attached to the CAT of *b*, & DO cc = [RETURN a list of *a* attached to the 1st element of ans, and the 2nd element of ans]].
%%% these are the CF and the CP.

/iv./(b) ELSE FAIL.

2. IF *a* is EQUAL to 'li', then

(a) SET ans to FILTER3 of *b*, and IF ans is not NULL, then

i. IF ans is a string, then RETURN ans.

ii. IF *b* has a REFERENCE value NIL and a TNS value: the imperfect, then

IF *b* has a MOOD value: subjunctive or homonymic with the subjunctive, then

IF *b* has the CATEGORY VA, then ASSIGN to *x* the CAT: PVB, & DO dd = [PASSON2 from *b* to *x* the property heritage: NGP, TRVY, & VOC, ASSIGN to *x* the TNS & MMK: imp, sub, & RETURN *x*].

IF *b* is homonymic for NUMBER between singular and plural and has the MMK: jis, then DO ee = [ASSIGN to *x* the NBR: pl, PASSON2 from *b* to *x* the property heritage: GDR, PRS, TRVY, & VOC, ASSIGN to *x* the TNS & MMK: imp, sub, & a CAT which is P attached to the CAT of *b*], then RETURN *x*.

ELSE DO dd.

iii. IF *b* has the TNS value: imperfect and *b* is not Annexed, then

IF *b* has a MOOD value: subjunctive or homonymic with the subjunctive,
IF *b* is homonymic for NUMBER between singular and plural and has the MMK: jis, then PASSON2 the DFN, CMK, & REF of *b* to *x*, then DO ee & cc.

ELSE ASSIGN to *x* the MMK: sub, & DO bb.

/iv./(b)/3./*STEP II./STEP C.* ELSE FAIL.

SCHEME 15: FUTURIZE: A Procedural Left-to-Right Scheme for Future CCF Recognition

ARGUMENTS: an inputword x and a rootlist.
OBJECTIVES:
EXTRACT a raw consonantal form from the inputword, then perform simple transformations on it, verify is MEMBERship in the rootlist, and IF so RETURN it as the value.
LOCAL VARIABLES: a, b, L, cc, and ss.
BASIC METHOD:
STEP I:

1. SET cc to be MKWORD of the INTERSECTION of the SCRIPTed inputword with the consonants,
2. SET ss to be the SCRIPTed cc, and
3. SET L to be the LENGTH of ss.

STEP II:
IF L is strictly LESS than 2, or IF L is GREATER than or EQUAL to 6, then FAIL.
STEP III:
IF cc is a MEMBER of the rootlist, and L is LESS than or EQUAL to 3, then RETURN cc.
STEP IV:
IF the initial Xer of ss is <m>, then DO: SET a to MKWORD of ss with the initial <m> DELETed, and

1. IF a is in the rootlist, then RETURN a.
2. IF the final Xer of ss is <t>, then DO: SET a to MKWORD of ss with the initial <m> and the final <t> DELETed, and proceed as in *STEP IV.* 1.
3. IF the penultimate Xer of ss is <:>, then DO: SET a to MKWORD of ss with the penultimate <:> DELETed, and proceed as in STEP IV. 1.

STEP V:
IF the final Xer of ss is <h>, then SET a to MKWORD of ss with the final <h> DELETed, and proceed as in *STEP IV.* 1.
STEP VI:
IF the 2nd and the 3rd Xers of ss are EQUAL, then

1. IF the final Xer of ss is <t>, then SET a to MKWORD of ss with the final <t> DELETed, and proceed as in *STEP IV.* 1.
2. IF the initial Xer of ss is <:>, then
   (a) Proceed as in *STEP IV.* 3.
   (b) ELSE SET a to MKWORD of ss with its 2nd Xer DELETed, and proceed as in *STEP IV.* 1.

STEP VII:
IF the penultimate Xer of ss is <:>, then DO: SET *a* to MKWORD of ss with the penultimate <:> DELETed, and

1. Proceed as in *STEP IV.* 1.

2. IF the final Xer of ss is <t>, then

   (a) IF the initial Xer of ss is <:>, then SET *a* to MKWORD of ss with the initial <:> and the final <t> DELETed, and proceed as in *STEP IV.* 1.

   (b)/3. ELSE FAIL.

STEP VIII:
IF the final Xer of ss is <:>, then DO: SET *a* to MKWORD of ss with the final <:> DELETed, and

1. Proceed as in *STEP IV.* 1.

2. IF the initial Xer of ss is <:>, then SET *a* to MKWORD of ss with the initial <:> DELETed, and

   (a) Proceed as in *STEP IV.* 1.

   (b) ELSE SET *a* to MKWORD of ss with the initial and final <:> DELETed, and proceed as in *STEP IV.* 1.

STEP IX:
IF the final Xer of ss is <t>, then

1. IF the initial Xer of ss is <:>, then SET *a* to MKWORD of ss with the initial <:> and the final <t> DELETed, and

   (a) Proceed as in *STEP IV.* 1.

   (b) IF the 2nd Xer of ss is <m>, then SET *a* to MKWORD of ss with the initial <:>, the 2nd <m> and the final <t> DELETed, and proceed as in *STEP IV.* 1.

2. ELSE proceed as in *STEP VI.* 1.

STEP X:
IF the initial Xer of ss is <:>, then

1. SET *a* to MKWORD of ss with the initial <:> DELETed, and proceed as in *STEP IV.* 1.

2. IF the 2nd Xer of ss is <m>, then SET a to MKWORD of ss with the initial <:> and the 2nd <m> DELETed, and proceed as in STEP IV. 1.

STEP XI:
IF the 2nd Xer of ss is <:>, then SET a to MKWORD of ss with the 2nd <:> DELETed, and proceed as in *STEP IV.* 1.
STEP XII:
IF the final Xer of ss is <n>, then SET a to MKWORD of ss with the final <n> DELETed, and proceed as in *STEP IV.* 1.
STEP XII: ELSE FAIL.

SCHEME 16: EXSTEM: A Procedural Scheme for Stem EXTRACTion

ARGUMENTS: an inputword x and a GPL y which is a set of association lists of roots and patterns.
OBJECTIVES:
obtain a valid Nominal DERIVation from the inputword x by EXTRACTing a Verbal root or a non-Verbal stem from it, retrieving the lists of patterns ASSOCIATed with the root or stem in the database, and scanning these lists for one pattern that MATCHes the inputword x. The DERIVation is from the Basic Pattern type for DERIVEA, and from the Broken Plural Pattern type for DERIVEB.
LOCAL VARIABLES: pp, root/stem, and ans.
BASIC METHOD: %%% DERIVEA.
STEP 1:
SET root/stem to be EXROOT/EXSTEM of the inputword x from the rootlist.
STEP 2:
IF the GPL y is empty, then FAIL.
STEP 3:
IF root/stem is not NULL, then SET pp to be the 1st element of the LPL obtained by ASSOCIATing root/stem with the GPL y, then
(a) IF pp MATCHes x, then RETURN a list which is the name of the LPL, root/stem, and pp. %%% using MATCHX.
(b) ELSE SET the GPL y to be the REST of y, then GOTO *STEP 2.*
BASIC METHOD: %%% DERIVEB.
STEP 1:
SET root/stem to be EXROOT/EXSTEM of the inputword x from the rootlist.
STEP 2:
IF the GPL y is empty, then FAIL.
STEP 3:
IF root/stem is not NULL, then SET pp to be the list of the REST of the LPL obtained by ASSOCIATing root/stem with the GPL y, then
(a) SET ans to MATCHW of pp and x, and IF ans is not NULL, then RETURN a list which is the name of the LPL, root/stem, and ans.
(b) ELSE SET the GPL y to be the REST of y, then GOTO *STEP 2.*

SCHEME 17: NDERIVE: An Iterative Scheme for Nominal DERIVation

ARGUMENTS: an inputword LL for Verbal or NN for (non-Verbal) Nominal.
OBJECTIVES:
† recognize a masculine singular Simple NF without CASE inflection for Sound Verbals or Nominals;
† recognize a masculine singular Simple NF with CASE inflection for Defective Weak Verbals or Nominals;
† the recognition referred to here is achieved by obtaining a valid DERIVation for the inputword and identifying Defective Weak suffixes if any.
LOCAL VARIABLES: *a*, *b*, *s*, NN, rr, and ans.
BASIC METHOD: %%% MKMASC1.
STEP A:
SET ans to be DERIVEP of LL from the GPL: "MLIST+", and IF ans is not NULL, then SET NN to be the 1st element of ans, then %%% ans is a list being NN, the LPL, the root, and the pattern.
DO aa = [ASSIGN to LL these NG & TNS values: (M) S, imp, PASSON to LL the CAT & VOC of NN, & the TRVY of the root, and RETURN a list of LL and ans].
STEP B:
SET ans to be DERIVEP of LL from the exceptional GPL: "XLIST1", and IF ans is not NULL, then SET NN to be the 1st and rr the 2nd elements of ans, then
STEP I:
IF rr is not in the Defective root set: {t-nn, bnn, gnn, d-kk, t-n, bn, gn, d-k, :x, m:, ns, svq, h-d}, then DO aa.
STEP II: ELSE FAIL.
STEP C:
ELSE SET *s* to be SEGMV 1 of LL, SET *a* to the 1st and *b* to the 2nd elements of *s*, then
STEP I:
IF *a* is the suffix <i+>, then

1. SET ans to be DERIVEP of *b* from "XLIST1", and IF ans is not NULL, then SET NN to be the 1st and rr the 2nd elements of ans, then

   (a) IF rr is in the Defective root set: {t-nn, bnn, gnn, d-kk, t-n, bn, gn, d-k, :x, m:, ns, svq, h-d}, then ASSIGN to ll these NGC & TNS values: (M) S, npm, imp, PASSON to LL the CAT & VOC of NN, the TRVY of rr, the TYP & FLXN of <i+>, and RETURN a list of LL and ans.

   (b)/2. ELSE FAIL.

STEP II:
ELSE SET *s* to be SEGMV 2 of LL, SET *a* to the 1st and *b* the 2nd elements of *s*, then

1. IF *a* is the Defective/plural suffix 'iy', then

   (a) SET ans to be DERIVEP of *b* from "XLIST1", and IF ans is not NULL, then SET NN to be the 1st and rr the 2nd elements of ans, then

      i. IF rr is in the Defective root set: {t-nn, bnn, gnn, d-kk, t-n, bn, gn, d-k, :x, m:, ns, svq, h-d}, then ASSIGN to ll these NGC, FLXN, TYP, TNS, HTY, & PLC: (M) SP, xnp, svm, fvc, imp, xh, rmx, PASSON to LL the CAT & VOC of NN, the TRVY of rr, and RETURN a list of LL and ans.

ii./(b)/2./*STEP III./STEP D.* ELSE FAIL.

BASIC METHOD: %%% MKMASC2.
STEP A:
SET ans to be DERIVEC of NN from the GPL: "MLIST+", and IF ans is not NULL, then
STEP I:
IF the 2nd element of ans is the stem 't-mn' and the penultimate Xer of NN is not <|>, then DO bb = [ASSIGN to NN the NG values: (M) S, PASSON to NN the CAT of the 1st element of ans, and RETURN a list of NN and ans]. %%% the 1st element of ans is the LPL.
STEP II: ELSE DO bb.
STEP III: ELSE FAIL.
STEP B:
ELSE SET *s* to be SEGMV 1 of NN, SET *a* to the 1st and *b* the 2nd elements of *s*, then
STEP I:
SET ans to be DERIVEC of *b* from the exceptional GPL: "XLISTK", then IF ans is not NULL, then

1. IF the 2nd element of ans is the stem 't-mn', then

   (a) IF *a* is the suffix <i+>, then ASSIGN to NN these NGC values: (M) S, npm, PASSON to NN the FLXN & TYP of <i+> and the CAT of the 1st element of ans, and RETURN a list of NN and ans.

   (b)/2. ELSE FAIL.

STEP II:
ELSE SET *s* to be SEGMV 2 of NN, SET *a* to the 1st and *b* the 2nd elements of *s*, then

1. IF *a* is the Defective suffix 'iy', then

   (a) SET ans to DERIVEC of *b* from "XLISTK" and IF ans is not NULL, then

      i. IF the 2nd element of ans is the stem 't-mn', then ASSIGN to NN the NGC, FLXN, & TYP values: (M) S, npm, svm, fvc, PASSON to NN the CAT of the 1st element of ans, and RETURN a list of NN and ans.

   ii./(b)/2./*STEP III./STEP C.* ELSE FAIL.

SCHEME 18: MKMASC: A Procedural Right-to-Left Scheme for Masculine Singular Recognition

ARGUMENTS: an inputword LL for Verbal or NN for (non-Verbal) Nominal.
OBJECTIVE:
recognize a feminine singular Simple NF without CASE inflection by obtaining a valid DERIVation for the inputword and identifying a feminine GENDER suffix if any.
LOCAL VARIABLES: *a*, *b*, *c*, *s*, NN, and ans.
BASIC METHOD: %%% FEMINIZE1.
STEP A:
SET ans to be DERIVEP of LL from the exceptional GPL: "XLIST3", and IF ans is not NULL, then
%%% "XLIST3" contains association lists for conventionally or sexually—as opposed to morphologically—feminine roots.
DO aa = [SET NN to be the 1st element of ans, ASSIGN to LL the NG & TNS values: (F) S, imp, PASSON to LL the CAT & VOC of NN, the TRVY of the 2nd element of ans, and RETURN a list of LL and ans].
STEP B:
ELSE SET *s* to SEGMV 2 of LL, SET *a* to the 1st and *b* the 2nd elements of *s*, then
STEP I:
IF *a* is the GENDER suffix: 'at', then

1. SET ans to be DERIVEP of *b* from the GPL: "!*FLIST", and IF ans is not NULL, then DO aa.

2./*STEP II./STEP C.* ELSE FAIL.

BASIC METHOD: %%% FEMINIZE2.
STEP A:
SET ans to be DERIVEC of NN from the exceptional GPL: "XLISTC", then IF ans is not NULL, then
STEP I:
IF the 2nd element of ans is the stem ':mm', then

1. IF the final Xer of NN is not <h>, then DO bb = [ASSIGN to ll the NG values: (F) S, PASSON to LL the CAT of the 1st element of ans, and RETURN a list of NN and ans].

2. ELSE FAIL.

STEP II:
IF the 2nd element of ans is in: {qdm, frsv}, then

1. IF the penultimate Xer of NN is not <|>, then DO bb.

2./*STEP III.* ELSE FAIL.

STEP B:
SET ans to be DERIVEC of NN from the exceptional GPL: "XLISTK", then IF ans is not NULL, then %%% "XLISTK" contains association lists for Df.Ab. Nominal stems.
STEP I:

IF the 2nd element of ans is in the Abbreviated stem set: {:l, sd, bd;, xd;r, h-mr, zrq, s;fr}, then DO bb.
STEP II: ELSE FAIL.
STEP C:
SET ans to be DERIVEC of NN from the exceptional GPL: "XLISTE", then IF ans is not NULL, then
STEP I:
IF the 2nd element of ans is the stem: 'h-d', then DO bb.
STEP II: ELSE FAIL.
STEP D:
SET *s* to SEGMV 2 of NN, SET *a* to the 1st and *b* to the 2nd elements of *s*, then
STEP I:
IF *a* is the GENDER suffix: 'ot', then

1. SET ans to be DERIVEC of *b* from the exceptional GPL: "XLISTC", then IF ans is not NULL, then

   (a) IF the 2nd element of ans is in the Defective stem set: {:b, :x}, then DO bb.

   (b)/2. ELSE FAIL.

STEP II:
IF *a* is the GENDER suffix: 'at', then

1. IF the stem of *b* is: 'bn', then %%% invoking EXSTEM.

   (a) IF the initial Xer of NN is <:>, then

      i. SET ans to be DERIVEC of *b* from "XLISTC", and IF ans is not NULL, then DO bb.

      ii. ELSE FAIL.

   (b) IF the final Xer of NN is <y>, then SET *c* to MKWORD of *b*, after SUBSTITUTing <y> with <:>, then

      i. SET ans to be DERIVEC of *c* from the exceptional GPL: "XLISTA", and IF ans is not NULL, then DO bb.

      ii./(c) ELSE FAIL.

2. ELSE SET ans to be DERIVEC of *b* from "XLISTA", then IF ans is not NULL, then

   (a) IF the 2nd element of ans is not the stem 'rkb', then DO bb.

   (b) ELSE FAIL.

3. ELSE SET ans to be DERIVEC of *b* from "XLISTC", then

   (a) IF the 2nd element of ans is the stem 'frsv', then

      i. IF the penultimate Xer of *b* is <y>, then DO bb.

      ii. ELSE FAIL.

   (b) IF the 2nd element of ans is not in the stem set: {qdm, ns, bn, :x}, then DO bb.

   (c) ELSE FAIL.

4. ELSE SET ans to be DERIVEC of *b* from the GPL: "FLIST", then IF ans is not NULL, then

   (a) IF the 2nd element of ans is in the Abbreviated stem set: {:l, sd, bd;, xd;r, h-mr, zrq, s;fr}, then

      i. IF the final Xer of *b* is not in: {:, y}, then DO bb.

      ii. ELSE FAIL.

   (b) ELSE DO bb.

   (c)/5./*STEP III./STEP E.* ELSE FAIL.

SCHEME 19: FEMINIZE: A Procedural Right-to-Left Scheme for Feminine Singular Recognition

---

ARGUMENTS: an inputword LL for Verbal or NN for (non-Verbal) Nominal.
OBJECTIVE:
recognize a dual Simple NF in the masculine or feminine category by identifying an obligatory dual NGC suffix and invoking MKMASC or FEMINIZE in order to obtain a valid DERIVation for, and identify any suffixes occurring in, the centre of the inputword.
LOCAL VARIABLES: *i*, *a*, *b*, *c*, *d*, *s*, ss, and ans.
BASIC METHOD: %%% DUALIZE.
STEP A:
IF the final Xer of LL/NN is in the set of dual suffix boundaries: {i, o, w, |}, then DO:
%%% entry condition.
STEP I:
SET *i* to 2, then
STEP II:
IF *i* is GREATER than or EQUAL to 6, then FAIL.
STEP III:
IF *i* is strictly LESS than 6, then SET *s* to be SEGMV of LL/NN using the index *i*, SET *a* to be the 1st and *b* the 2nd elements of *s*, then %%% *a* is the suffix, *b* the centre.

1. IF *a* is in the set of dual suffixes, then

   (a) SET ans to be MKMASC1 of *b*, and %%% DUALIZE1.
   IF ans is not NULL, then

      i. IF *b* is not carrying full vowel mark, then
      %%% this means that *b* is not an inflected Defective.
      ASSIGN to LL the NG & TNS values: (M) D, imp, PASSON to ll the CMK & FLXN of *a*, the CAT, VOC, & TRVY of *b*, and RETURN a list of LL and the REST of ans.
      %%% the REST here is: the LPL, the root, and the pattern.

      ii. ELSE FAIL.

   (b) ELSE SET ans to be FEMINIZE1 of *b*, and IF ans is not NULL, then ASSIGN to LL the GDR & TNS values: ff, imp, PASSON to ll the NBR, CMK, & FLXN of *a*, the CAT, VOC, & TRVY of *b*, and RETURN a list of LL and the REST of ans.

   (c) ELSE FAIL.

   %%% DUALIZE2.

2. ELSE SET ans to be DERIVEC of *b* from the exceptional GPL: "XLISTL",
   %%% "XLISTL" contains dual-only association lists.
   and IF ans is not NULL, then DO aa = [ASSIGN to NN the GDR: mc, PASSON to NN the NBR, CMK, & FLXN of *a*, the CAT of the 1st element of ans, and RETURN a list of NN and ans].

3. ELSE SET ans to be DERIVEC of *b* from the exceptional GPL: "XLISTJ", and IF ans is not NULL, then %%% "XLISTJ" is an MF-Numeral association list.

   (a) IF the 2nd element of ans is the stem 'h-d', then DO aa.

   (b) ELSE FAIL.

4. ELSE SET ans to be DERIVEC of *b* from the exceptional GPL: "XLISTK", and IF ans is not NULL, then %%% "XLISTK" contains Numerals.

   (a) IF the 2nd element of ans is the stem ':lf', then DO aa.

   (b) ELSE FAIL.

5. ELSE IF the stem of *b* is in the Defective set: {:b, :x}, then %%% using EXSTEM.

   (a) IF the final 2 Xers of *b* are 'aw', then SET *c* to be *b* with the final 'aw' DELETed, then

      i. SET ans to be DERIVEC of *c* from the exceptional GPL: "XLISTB", and IF ans is not NULL, then DO aa.

ii. ELSE FAIL.

(b) ELSE SET ans to be FEMINIZE2 of *b* and IF ans is not NULL, then DO bb = [ASSIGN to NN the GDR: ff, PASSON to NN the NBR, CMK, & FLXN of *a*, the CAT of *b*, and RETURN a list of NN and the REST of ans].

(c) ELSE FAIL.

6. ELSE IF the stem of *b* is in the Defective set: {sd, bd;, xd;r, h-mr, s;fr, zrq}, then

(a) IF the final Xer of *b* is <w>, then SET *c* to MKWORD of *b* after SUBSTITUTing <w> with <:>, then

i. SET ans to be FEMINIZE2 of *c*, and IF ans is not NULL, then ASSIGN to NN the GDR: ff, PASSON to NN the NBR, CMK, & FLXN of *a*, the CAT of *c*, and RETURN a list of NN and the REST of ans.

ii. ELSE FAIL.

(b) IF the final Xer of *b* is <:>, then

i. SET ans to be DERIVEC of *b* from the exceptional GPL: "XMLIST", then %%% "XMLIST" does not contain Numerals.
IF ans is not NULL, then DO aa.

ii. ELSE SET ans to be FEMINIZE2 of *b*, and IF ans is not NULL, then DO bb.

iii./(c) ELSE FAIL.

7. ELSE SET ans to be DERIVEC of *b* from "XMLIST", and IF ans is not NULL, then DO aa.

8. ELSE SET ans to be FEMINIZE2 of *b*, and IF ans is not NULL, then SET *c* to be the 1st and *d* the 2nd element of ans, then

(a) IF the 2nd element of *d* is the stem 'h-d', then

i. IF the final Xer of *b* is not <y>, then DO bb.

ii. ELSE FAIL.

(b) ELSE DO bb.

(c) ELSE FAIL.

9. ELSE SET ss to be SEGMV 2 of *b*, SET *c* to the 1st and *d* the 2nd elements of ss, then

(a) IF *c* is the GENDER suffix 'at', then

i. SET ans to be DERIVEC of *d* from "XLISTL", and IF ans is not NULL, then ASSIGN to NN the GDR: ff, the NBR, CMK, & FLXN of *a*, the CAT of the 1st element of ans, and RETURN a list of NN and ans.

ii./(b)/10. ELSE FAIL.

STEP IV:
ELSE increment *i* by 1 and GOTO *STEP II.* %%% DUALIZE1 and 2.
STEP V./STEP B. ELSE FAIL.

SCHEME 20: DUALIZE: An Iterative Right-to-Left Scheme for Dual Recognition

---

ARGUMENTS: an inputword LL for Verbal or NN for (non-Verbal) Nominal.
OBJECTIVE:
recognize a Masculine Regular Plural Simple NF by identifying an obligatory plural NGC suffix and invoking MKMASC in order to obtain a valid DERIVation for, and identify any suffixes occurring in, the centre of the inputword.
LOCAL VARIABLES: *i*, *a*, *b*, *c*, *s*, and ans.
BASIC METHOD:
STEP A:
IF the final Xer of LL/NN is in the set of plural suffix boundaries: {a, o, w, y}, then DO:
%%% entry condition.
STEP I:
SET *i* to 1, then
STEP II:
IF *i* is GREATER than or EQUAL to 5, then FAIL.
STEP III:
IF *i* is strictly LESS than 5, then DO: SET *s* to be SEGMV *i* of LL/NN, SET *a* to be the 1st and *b* the 2nd elements of *s*, then %%% MPLURALIZE1.

1. IF *a* is in the set of Df.Ab. suffix allomorphs: {o, ona}, then

   (a) SET ans to DERIVEP of *b* from the exceptional GPL: "XLIST2", and IF ans is not NULL, then ASSIGN to LL these NGC, HTY, PLC, & TNS: (M) P, cpm, hm, rmp, imp, PASSON to LL the FLXN of *a*, the CAT & VOC of the 1st element of ans, the TRVY of its 2nd element, and RETURN a list of LL and ans.

   (b) ELSE IF the final Xer of *b* is <w>, then SET *c* to MKWORD of *b*, after SUBSTITUTing <w> with <y>, then

      i. SET ans to DERIVEP of *c* from "XLIST2", then ASSIGN to LL these NGC, FLXN, HTY, PLC, & TNS values: (M) P, nmm, hm, rmp, imp, PASSON to LL the FLXN of *a*, the CAT & VOC of the 1st element of ans, the TRVY of its 2nd element, and RETURN a list of LL and ans.

      ii./(c) ELSE FAIL.

2. IF *a* is in the set of MRP suffixes, then

   (a) SET ans to MKMASC1 of *b*, and IF ans is not NULL, then

      i. IF *b* is not Defective, then ASSIGN to LL the NG & TNS values: (M) P, imp, PASSON to LL the CMK, FLXN, PLC, & HTY of *a*, and the CAT, VOC, & TRVY of *b*, and RETURN a list of LL and the REST of ans.

      ii. ELSE FAIL.

   (b) IF *a* is not the Defective suffix 'iy', then

      i. SET ans to DERIVEP of *b* from the exceptional GPL: "XLIST1", and IF ans is not NULL, then

         IF the 2nd element of ans is in the Df.Wk root set: {t-nn, bnn, gnn, d-kk, t-n, bn, gn, d-k, ;x, m:, ns, svq, h-d}, then ASSIGN to LL the NG & TNS values: (M) P, imp, PASSON to LL the CMK, FLXN, PLC, & HTY of *a*, the CAT & VOC of the 1st element of ans, the TRVY of its 2nd element, and RETURN a list of LL and ans.

         /ii./(c) ELSE FAIL.

3. /

STEP IV: %%% MPLURALIZE2.
IF *a* is in the set of MRP suffixes, then

1. IF the stem of *b* is 'bn', then %%% invoking EXSTEM.

   (a) IF the 1st two Xers of *b* are 'ba', then SET *c* to MKWORD of *b* after SUBSTITUTing <a> with <i>, then

      i. SET ans to DERIVEC of *c* from the exceptional GPL: "XLISTC", and IF ans is not NULL, then DO aa = [ASSIGN to NN the NG values: (M) P, PASSON to NN the CMK, FLXN, PLC, & HTY of *a*, the CAT of the 1st element of ans, and RETURN a list of NN and ans].

      ii./(b) ELSE FAIL.

2. ELSE SET ans to DERIVEC of *b* from the exceptional GPL: "XLISTI", and IF ans is not NULL, then

   (a) IF the 2nd element of ans is not in the stem set: {sd, gn, svq, d-k, bd;, fqr, t-mn, h-sn, nfd-, :mm, d-hb, xd;r, h-mr, s;fr, zrq}, then DO aa.

   (b) ELSE FAIL.

3. ELSE IF *i* is EQUAL to 4, then

(a) IF the stem of *b* is 'εsvr', then

i. IF the 2nd Xer of *b* is <i>, then

SET ans to DERIVEC of *c* from "XLISTK", and IF ans is not NULL, then DO bb = [ASSIGN to NN the NG values: (M) P, PASSON to NN the CMK, FLXN, & PLC of *a*, the CAT of the 1st element of ans, and RETURN a list of NN and ans].

/ii. ELSE FAIL.

(b) ELSE SET ans to DERIVEC of *b* from "XLISTK", and IF ans is not NULL, then

(c) IF the 2nd element of ans is the stem 't-mn', then DO bb.

(d) ELSE FAIL.

(e) ELSE SET ans to DERIVEC of *b* from "XLISTJ", and IF ans is not NULL, then

i. IF the 2nd element of ans is the stem 'h-d', then DO aa.

ii. ELSE DO bb.

iii./(d) ELSE FAIL.

4. ELSE increment *i* by 1 and GOTO *STEP II.*

STEP V: ELSE FAIL.

SCHEME 21: MPLURALIZE: An Iterative Right-to-Left Scheme for Masculine Regular Plural Recognition

---

ARGUMENTS: an inputword LL for Verbal or NN for (non-Verbal) Nominal.
OBJECTIVE:
recognize a Feminine Regular Plural Simple NF without CASE inflection by obtaining a valid DERIVation for the centre of the inputword and identifying an obligatory FRP GENDER suffix.
LOCAL VARIABLES: *a*, *b*, *c*, *s*, ss, and ans.
BASIC METHOD: %%% FPLURALIZE1.
STEP A:
DO aa = [SET *s* to SEGMV 3 of LL, SET *a* to the 1st and *b* the 2nd elements of *s*], then
STEP I:

IF *a* is the GENDER suffix 'a|t', then

1. SET ans to DERIVEQ of *b* from the exceptional GPL: "XLIST3", and IF ans is not NULL, then DO bb = [PASSON to LL the NG, PLC, & HTY of *a*, the CAT & VOC of the 1st element of ans, the TRVY of its 2nd element, ASSIGN to LL the TNS: imp, and RETURN a list of LL and ans].

2. ELSE SET ans to DERIVEP of *b* from the GPL: "!*FLIST", and IF ans is not NULL, then DO bb.

3./*STEP II./STEP B.* ELSE FAIL.

BASIC METHOD: %%% FPLURALIZE2.
STEP A:
DO aa, then
STEP I:
IF *a* is the GENDER suffix 'a|t', then

1. SET ans to DERIVEC of *b* from the exceptional GPL: "XLISTA", and IF ans is not NULL, then

   (a) IF the 2nd element of ans is in the stem set: {h-ml, :mm, h-mr, rkb, ktb, mnh-}, then DO cc = [PASSON to NN the NG, PLC, & HTY of *a*, the CAT of the 1st element of ans, and RETURN a list of NN and ans].

   (b) ELSE FAIL.

2. SET ans to DERIVEC of *b* from the exceptional GPL: "XLISTG", and IF ans is not NULL, then

   (a) IF the 2nd element of ans is in the stems: {kn, bt}, then DO dd = [PASSON to NN the NG & PLC of *a*, the CAT of the 1st element of ans, and RETURN a list of NN and ans].

   (b) ELSE FAIL.

3. SET ans to DERIVEC of *b* from the exceptional GPL: "XLISTC", and IF ans is not NULL, then

   (a) IF the 2nd element of ans is in the stems: {frsv, h-dq}, then

      i. IF the penultimate Xer of *b* is not <y>, then DO dd.

      ii. ELSE FAIL.

   (b) IF the 2nd element of ans is in the stems: {ld, :mm}, then

i. IF either the final Xer of *b* is <h> or the penultimate Xer of *b* is <y>, then DO cc.

ii. ELSE DO dd.

iii. ELSE FAIL.

(c) IF the 2nd element of ans is in the stems: {ɛmm, s;dq}, then

i. IF the penultimate Xer of *b* is <|>, then DO dd.

ii. ELSE DO bb.

iii. ELSE FAIL.

(d) IF the 2nd element of ans is in the stems: {sd, xl, jls, nzl, d/nn, ɛsvr, jdd, :lf, rkb}, then DO cc.

(e) IF the 2nd element of ans is in the stems: {rd, qm, bd;, sbɛ, ɛlm, svrf, svbk, qs;r, ktb, drs}, then DO dd.

(f) ELSE FAIL.

4. IF the stem of *b* is 'bn', then %%% using EXSTEM.

(a) IF the 1st two Xers of *b* are 'ba', then SET *c* to MKWORD of *b* after SUBSTITUTing <a> with <i>, then

i. SET ans to DERIVEC of *c* from "XLISTC", and IF ans is not NULL, then DO cc.

ii. ELSE FAIL.

(b) IF the final Xer of *b* is <y>, then SET *c* to MKWORD of *b* after SUBSTITUTing <y> with <:>, then

i. SET ans to DERIVEC of *c* from "XLISTA", and IF ans is not NULL, then DO dd.

ii./(c) ELSE FAIL.

5. IF the stem of *b* is ':x', then %%% using EXSTEM.

(a) IF the 2nd Xer of *b* is <a> and the final two Xers of *b* are 'aw', then SET *c* to MKWORD of *b* after SUBSTITUTing the 1st <a> with <u> and DELETing the final 'aw', then

i. SET ans to DERIVEC of *c* from "XLISTC", and IF ans is not NULL, then DO cc.

ii./(b) ELSE FAIL.

6. IF the stem of *b* is 'h-d', then

   (a) IF the 2nd Xer of *b* is <a>, then SET *c* to MKWORD of *b* after SUBSTITUTing the 1st <a> with <i> and the 2nd <a> in *b* with <o>, then

      i. SET ans to DERIVEC of *c* from "XLISTC", and IF ans is not NULL, then DO dd.

      ii. ELSE SET ans to DERIVEC of *b* from the GPL: "FLIST", and IF ans is not NULL, then DO cc.

      iii. ELSE FAIL.

   (b) IF the final Xer of *b* is not <y>, and the 3rd Xer of *b* is not <|>, then

      i. SET ans to DERIVEC of *b* from "FLIST", and IF ans is not NULL, then DO cc.

      ii./(c) ELSE FAIL.

7. IF the stem of *b* is in the set: {sd, bd;, xd;r, h-mr, zrq, s;fr}, then

   (a) IF the final Xer of *b* is <w>, then SET *c* to MKWORD of *b* after SUBSTITUTing <w> with <:>, then

      i. SET ans to DERIVEC of *c* from "XLISTD", and IF ans is not NULL, then DO cc.

      ii. ELSE FAIL.

   (b) IF the final Xer of *b* is not <:>, then

      i. SET ans to DERIVEC of *b* from "FLIST", and IF ans is not NULL, then DO dd.

      ii./(c) ELSE FAIL.

8. ELSE SET ans to DERIVEC of *b* from "FLIST", and IF ans is not NULL, then

   (a) IF the LPL is in: {ADLIST1, ADLIST3}, then

      i. IF the 2nd element of ans is in the set: {s:, d-hb, nfd-, t-mn}, then DO dd.

      ii. ELSE DO cc.

      iii. ELSE FAIL.

   (b) ELSE DO dd.

(c)/9./*STEP II./STEP B.*. ELSE FAIL.

SCHEME 22: FPLURALIZE: A Procedural Right-to-Left Scheme for Feminine Regular Plural Recognition

---

ARGUMENTS: an inputword NN for Nominal.
OBJECTIVE:
recognize a Broken Plural Simple NF in the masculine or feminine category without CASE inflection for Sound Nominals and with CASE inflection for Defective Weak Nominals by obtaining a valid DERIVation for the centre of, and identifying any Defective Weak suffixes occurring in, the inputword.
LOCAL VARIABLES: *a*, *b*, *s*, and ans.
BASIC METHOD:
STEP I:
SET ans to DERIVEZ of NN from the GPL: "MDLIST", and IF ans is not NULL, then ASSIGN to NN the NG, HTY, & CAT: (M) P, hm, NN, and RETURN a list of NN and ans.
STEP II:
SET ans to DERIVEZ of NN from the GPL: "FDLIST", and IF ans is not NULL, then

1. IF the 2nd element of ans is in the Defective stem set: {ns, bn, gn, t-n, svq}, then ASSIGN to NN the NG, HTY, & CAT: (F) P, hm, NN, and RETURN a list of NN and ans.

2. ELSE FAIL.

STEP III:
SET ans to DERIVEZ of NN from "XLISTA", and IF ans is not NULL, then

1. IF the 2nd element of ans is in the stem set: {h-ml, :mm}, then DO aa = [ASSIGN to NN the NG & HTY: (MF) P, hm, PASSON to NN the CAT of the 1st element of ans, and RETURN a list of NN and ans].

2. ELSE DO bb = [ASSIGN to NN the NG: (MF) P, PASSON to NN the CAT of the 1st element of ans, and RETURN a list of NN and ans].

3. ELSE FAIL.

STEP IV:
SET ans to DERIVEZ of NN from "XLISTB", and IF ans is not NULL, then

1. IF the 2nd element of ans is in the stem set: {:b, :x, xl, qm, sd, ld, bn, jls, d/nn, rjl, rkb, :lf, nzl, ɛsvr, d;rb, s;dq, ɛmm, jdd}, then DO cc = [ASSIGN to NN the NG & HTY: (M) P, hm, PASSON to NN the CAT of the 1st element of ans, and RETURN a list of NN and ans].

2. ELSE DO dd = [ASSIGN to NN the NG: (M) P, PASSON to NN the CAT of the 1st element of ans, and RETURN a list of NN and ans].

3. ELSE FAIL.

STEP V:
SET ans to DERIVEZ of NN from "XLISTC", and IF ans is not NULL, then

1. IF the 2nd element of ans is in the stem set: {ns, ld, :lf, εsvr, :mm}, then DO ee = [ASSIGN to NN the NG & HTY: (F) P, hm, PASSON to NN the CAT of the 1st element of ans, and RETURN a list of NN and ans].

2. ELSE DO ff = [ASSIGN to NN the NG: (F) P, PASSON to NN the CAT of the 1st element of ans, and RETURN a list of NN ans].

3. ELSE FAIL.

STEP VI:
SET ans to DERIVEZ of NN from the exceptional GPL: "XLISTF", and IF ans is not NULL, then DO bb.
STEP VII:
SET ans to DERIVEZ of NN from "XLISTG" or "XLISTK", and IF ans is not NULL, then

1. IF the 2nd element of ans is not in the Defective stem set: {bn, t-n, gn}, then DO dd.

2. ELSE FAIL.

STEP VIII:
SET ans to DERIVEZ of NN from "XLISTH", and IF ans is not NULL, then

1. IF the 2nd element of ans is in the stem set: {kbr, s;gr}, then DO ff.

2. ELSE DO ee.

STEP IX:
SET ans to DERIVEZ of NN from "XLISTI", and IF ans is not NULL, then

1. IF the 2nd element of ans is the stem 'svrf', then DO cc.

2. IF the 2nd element of ans is in the stem set: {d-hb, t-mn, nfd-}, then DO bb.

3. ELSE DO aa.

STEP X:
ELSE SET *s* to SEGMV 1 of NN, SET *a* to the 1st and *b* the 2nd elements of *s*, then

1. IF *a* is the suffix <i+>, then

   (a) SET ans to DERIVEZ of *b* from "FDLIST", and IF ans is not NULL, then

      i. IF the 2nd element of ans is in the Defective stem set: {ns, bn, t-n, gn, svq}, then ASSIGN to NN the NGC, HTY & CAT: (F) P, npm, hm, NN, PASSON to NN the FLXN & TYP of *a*, and RETURN a list of NN and ans.

      ii. ELSE SET ans to DERIVEZ of *b* from the exceptional GPL: "XLISTG", and IF ans is not NULL, then

         IF the 2nd element of ans is in the set: {bn, t-n, gn}, ASSIGN to NN the NGC: (M) P, npm, PASSON to NN the FLXN & TYP of *a*, the CAT of the 1st element of ans, and RETURN a list of NN and ans.

         /iii. ELSE FAIL.

   (b) ELSE SET *s* to SEGMV 2 of NN, SET *a* to the 1st and *b* the 2nd elements of *s*, then

      i. IF *a* is the Defective suffix 'iy', then

         SET ans to DERIVEZ of *b* from "FDLIST", and IF ans is not NULL, then

         IF the 2nd element of ans is in the Defective set: {ns, bn, t-n, gn, svq}, then ASSIGN to NN the NGC, FLXN, HTY, CAT, & TYP: (F) P, npm, svm, hm, NN, fvc, and RETURN a list of NN and ans.

         ELSE FAIL.

         SET ans to DERIVEZ of *b* from "XLISTG", and IF ans is not NULL, then

         IF the 2nd element of ans is in the Defective set: {bn, t-n, gn}, then ASSIGN to NN the NGC, FLXN, & TYP: (M) P, npm, svm, fvc, PASSON to NN the CAT of the 1st element of ans, and RETURN a list of NN and ans.

         /iii./(c)/2./*STEP XI./STEP B.* ELSE FAIL.

SCHEME 23: BPLURALIZE: A Procedural Right-to-Left Scheme for Broken Plural Recognition

---

ARGUMENTS: an inputword NN for Nominal.
OBJECTIVE:
MODIFY the NUMERICAL and HUMANITY values ASSIGNed in NPARSE taking into account CATEGORY and NUMBER contexts.
LOCAL VARIABLES: *m*, pp, rr, and ans.

BASIC METHOD:
STEP A:
SET ans to NPARSE of NN, and IF ans is not NULL, SET rr to the 3rd and pp the 4th elements of ans, then %%% rr is the stem and pp the pattern.
STEP I:
IF NN is a Numeral, then

1. IF NN is singular, then

   (a) IF the initial Xer of NN is <w>, then ASSIGN to NN the MRK: mk, the VAL of rr, and RETURN ans.

   (b) ELSE RETURN ans.

2. IF NN is dual, then SET *m* to the VAL of rr, then

   (a) IF *m* is GREATER than 10, then ASSIGN to NN the VAL of *m* multiplied by 2, and RETURN ans.

   (b) ELSE RETURN ans.

3. IF NN is plural, then

   (a) IF NN is masculine, then

      i. IF rr is 'ɛsvr', then ASSIGN to NN the VAL 20, and RETURN ans.

      ii. IF rr is not in the stem set: {h-d, :lf}, then ASSIGN to NN the VAL of rr multiplied by 10, and RETURN ans.

   iii./(b) ELSE RETURN ans.

4. ELSE FAIL.

STEP II:
IF NN is a Noun, then

1. IF NN is singular, then

   (a) IF NN is masculine and pp is 'caciyc' and rr is 'd;rb', then ASSIGN to NN the HTY: hp, and RETURN ans.

   (b) IF either pp is 'caciyc' and rr is in the stem set: {h-ml, jls, d/nn, jbl, :lf, nzl, s;dq, ɛsvr}, or pp is 'cacc' and rr is in: {ɛmm, jdd}, or pp is 'ca|c' and rr is 'xl', or pp is 'cayyic' and rr is 'sd', or pp is 'cacuc' and rr is 'rjl', or pp is in: {waciyc wacac} and rr is 'ld', or pp is in: {cacoc :icocac} and rr is 'mr:', or pp is in: {cica|c cucc} and rr is ':mm', or pp is in: {:icoc cic} and rr is 'bn', or pp is in: {cac cuc} and rr is in: {:b, :x}, then ASSIGN to NN the HTY: hp, and RETURN ans.

(c)/2./*STEP III.* ELSE RETURN ans.

STEP B. ELSE FAIL.

SCHEME 24: MVACT: A Procedural Scheme for Adjusting NUMERICAL and HUMANITY Values

---

ARGUMENTS: an inputword LL for Verbal.
OBJECTIVE:
MODIFY the TRANSITIVITY and HUMANITY values ASSIGNed in LPARSE taking into account NUMBER and VOICE contexts.
LOCAL VARIABLES: rr and ans.
BASIC METHOD:
STEP A:
SET ans to LPARSE LL, and IF ans is not NULL, then SET rr to the 3rd element of ans, then
STEP I:
IF LL is active, then

1. IF rr is 'h-sb', then

   (a) IF LL is singular, then ASSIGN to LL the HTY = hp, and DO aa = [ASSIGN to LL the TRVY: mxtr, and RETURN ans].

   (b) ELSE DO aa.

2. IF rr is 't-mn', then

   (a) IF LL is singular, then ASSIGN to LL the HTY = hp, and DO bb = [ASSIGN to LL the TRVY: mtr, and RETURN ans].

   (b) ELSE DO bb.

3. IF LL has the NBR value: sn, then ASSIGN to LL the HTY: hp, and RETURN ans.

4. ELSE RETURN ans.

STEP II:
IF LL is passive, then

1. IF rr is 'h-sb', then ASSIGN to LL the TRVY: nctr, and RETURN ans.

2. DEMOTE the TRVY value of rr & ASSIGN it to LL and RETURN ans.

3. ELSE RETURN ans.

STEP III./STEP B. ELSE FAIL.

SCHEME 25: TVACT2: A Procedural Scheme for Adjusting TRANSITIVITY and HUMANITY Values

---

ARGUMENTS: an inputword x for Verbal or other Nominals.
OBJECTIVE:
recognize a singular, dual or plural Simple NF with a CASE inflection and implement constraints on Diptotes of Type 1 and 2.
LOCAL VARIABLES: a, b, c, s, pp, and ans.
BASIC METHOD:
STEP A:
SET ans to MVACT of x, and IF ans is NULL, then SET ans to TVACT2 of x, and IF ans is not NULL, then SET s to SEGMV 2 x, SET a to the 1st and b to the 2nd elements of s, then
STEP I:
IF a is the Abbreviated suffix 'ay', then

1. IF x is a Verbal2, then ASSIGN to x the CMK & FLXN: ncp, svm, and RETURN x.

2. ELSE ASSIGN to x the CMK & FLXN: ncp, xvm, and RETURN x.

3. ELSE FAIL.

STEP II:
IF x has a full NGC suffix, then RETURN x.
STEP III:
IF x is a MRP, then

1. IF x is a Verbal1 or a Verbal2, then

   (a) IF a is the Defective suffix 'iy', then ASSIGN to x the NBR, HTY, DFN, REF, & CAT: sp, xh, xdf, xxl, WN, and RETURN a list of b and a.

   (b)/2. ELSE RETURN x.

STEP IV:
IF x is ambiguous between a MRP and a Defective, then ASSIGN to x the DFN: xdf, and RETURN x.
STEP V:

IF x has a reduced NGC suffix, then RETURN x.
STEP VI:
IF x is a Defective, then

1. IF x has a full vowel mark, then ASSIGN to x the DFN: ndf, and RETURN x.

2. IF x has the FLEXION of 'iy', then RETURN x.

3./*STEP VII.* ELSE FAIL.

STEP B:
ELSE SET a to the final Xer of x, SET b to the 2nd element of SEGMV 1 of x, then
STEP I:
IF b is in the set of Proper Nouns and the final Xer of b is not 'y', then SET c to the 1st element of SEGMV 2 of b, then

1. IF either c is the GENDER suffix 'at' or b MATCHes the pattern 'cawa|cic', then

   (a) IF a is <a>, then PASSON to x the NG of b, ASSIGN to x the CMK, FLXN, & DFN: cpm, svm, dfp, and RETURN x.

   (b) IF a is <u>, then PASSON to x the NG of b, ASSIGN to x the CMK, FLXN, & DFN: nmm, svm, dfp, and RETURN x.

   (c) ELSE FAIL.

2. IF a is a full vowel mark, then PASSON to x the NG of b, the CMK of a, ASSIGN to x the FLXN & DFN: svm, dfp, and RETURN x.

3. ELSE FAIL.

STEP II:
SET ans to MVACT b, and IF ans is NULL, then SET ans to TVACT2 b, and IF ans is not NULL, then

1. IF b is Defective and has the suffix 'iy', then

   (a) IF b is a Numeral, then

      i. IF a is <a>, then PASSON2 to x the NG, VAL, MRK, & CAT of b, PASSON to x the CMK & FLXN of a, and RETURN x.

      ii. IF a is <a+>, then PASSON2 to x the NG, VAL, MRK, & CAT of b, ASSIGN to x the CMK, FLXN, & DFN: acm, fvm, ndf, and RETURN x.

iii. ELSE FAIL.

(b) IF *b* is a Verbal1, then

i. IF *a* is <a>, then ASSIGN to *x* the NG: (M) S, PASSON2 to *x* the TRVY & VOC of *b*, PASSON to *x* the CMK & FLXN of *a*, and RETURN *x*.

ii. IF *a* is <a+>, then ASSIGN to *x* the NG: (M) S, PASSON2 to *x* the TNS, TRVY, VOC, & CAT of *b*, ASSIGN to *x* the CMK, FLXN, HTY, & DFN: acm, fvm, hp, ndf, and RETURN *x*.

iii. ELSE FAIL.

(c) IF *b* is a Noun, then

i. IF *a* is <a>, then PASSON2 to *x* the property heritage 1 = [NG, VAL, MRK, HTY, & CAT of *b*], PASSON to *x* the CMK & FLXN of *a*, ASSIGN to *x* the TYP: rnc, and RETURN *x*.

ii./(d) ELSE FAIL.

2. IF *b* has no CASE and no FLEXION, then

(a) IF *b* is a FRP, then

i. IF *a* is <i>, then

IF *b* is either a Verbal1 or a Verbal2, then PASSON2 to *x* the property heritage 2 = [the NG, TRVY, VOC, HTY, & CAT of *b*], and DO aa = [ASSIGN to *x* the CMK & FLXN: cpm, svm, and RETURN *x*].

ELSE PASSON2 to *x* the property heritage 1, and DO aa.

ELSE FAIL.

ii. IF *a* is <i+>, then

IF *b* is either a Verbal1 or a Verbal2, then PASSON2 to *x* the property heritage 3 = [the NG, TNS, TRVY, VOC, HTY, & CAT of *b*], and DO bb = [ASSIGN to *x* the CMK, FLXN, & DFN: cpm, fvm, ndf, and RETURN *x*].

ELSE PASSON2 to *x* the property heritage 1, and DO bb.

ELSE FAIL.

iii. IF *a* is <u>, then

IF *b* is either a Verbal1 or a Verbal2, then PASSON2 to *x* the property heritage 2, and DO cc = [PASSON to *x* the CMK & FLXN of *a*, and RETURN *x*].

ELSE PASSON2 to *x* the property heritage 1, and DO cc.

ELSE FAIL.

iv. IF *a* is <u+>, then

IF *b* is either a Verbal1 or a Verbal2, then PASSON2 to *x* the property heritage 3, and DO dd = [ASSIGN to *x* the CMK, FLXN, & DFN: nmm, fvm, ndf, and RETURN *x*].

ELSE PASSON2 to *x* the property heritage 1, and DO dd.

/v. ELSE FAIL.

(b) ELSE SET pp to be the 4th element of ans, SET *c* to be the 1st element of SEGMV 3 of *b*, then %%% pp is the pattern.

i. IF pp is in the Diptote 1 patterns: {maca|cic, :ama|cic, maca|yic, mawa|cic, cawa|cc, :aca|cic, maca|ciyc, caca|ciyc, :acoyac, :acowac, cawa|cic, cawa|:ic}, then

IF *a* is a single vowel mark, then DO ee = [PASSON2 to *x* the property heritage 1, ASSIGN to *x* the TYP: rnc, and DO cc].

ELSE FAIL.

ii. IF *c* is the Extended suffix 'a|:', and pp is not in the pattern set: {ca|ca|:, cica|:}, then

IF *a* is a single vowel mark, then DO ee.

ELSE FAIL.

iii. IF pp is ':acocac', and *b* is not a Numeral, then

IF *a* is a single vowel mark, then DO ee.

ELSE FAIL.

iv. IF *a* is a single vowel mark, then

IF *b* is either a Verbal1 or a Verbal2, then PASSON2 to *x* the property heritage 2, and DO cc.

ELSE PASSON2 to *x* the property heritage 1, and DO cc.

ELSE FAIL.

v. IF *a* is a full vowel mark, then

IF *b* is either a Verbal1 or a Verbal2, then PASSON2 to *x* the property heritage 3, and DO ff = [ASSIGN to *x* the CMK of *a*, the FLXN & DFN: fvm, ndf, and RETURN *x*].

ELSE PASSON2 to *x* the property heritage 1, and DO ff.

/vi./(c)/3./*STEP III./STEP C.* ELSE FAIL.

SCHEME 26: FLEX: A Procedural Right-to-Left Scheme for the Recognition of CASE-Inflected Simple NFs

---

ARGUMENTS: an inputword x for CNF.
OBJECTIVES:
† recognize a Complex NF which is a Simple NF attached with an optional Determiner.
† disambiguate some property values for NGC, FLXN, DFN, and CAT.
LOCAL VARIABLES: i, a, b, c, d, ss, and ans.
BASIC METHOD:
STEP A:
IF the final Xer of x is in the set of DNF boundaries: {a, i o, u, w, y, |, u+, a+, i+}, then DO:
STEP I:
IF x is in SPs or in DPs, then ASSIGN to x the DFN: dfp, and RETURN x.
STEP II:
IF x is in the set of Proper Nouns, then

1. IF x ends in <y>, then ASSIGN to x the DFN & CMK: dfp, ncp, and RETURN x.

2. ELSE FAIL.

STEP III:
ELSE SET ans to FLEX x, and IF ans is not NULL, then

1. IF x has no DFN value, ASSIGN to x the DFN: ndf, and RETURN x.

2. IF x is ambiguous in DFN, then

   (a) IF x is Defective, then RETURN ans.

   (b) ELSE ASSIGN to x the CMK: xxp, and RETURN ans.

   (c) ELSE FAIL.

3. ELSE RETURN ans.

STEP IV: ELSE FAIL.
STEP B:
IF the initial Xer of x is in the set of DNF boundaries: {:, l}, then

STEP I:
SET *i* to 4, then

1. IF *i* is strictly LESS than **1**, then FAIL.

2. IF *i* is GREATER than 0, then DO: SET ss to SEGM *i* of *x*, SET *a* to the 1st and *b* to the 2nd elements of ss, then %%% *a* is the Determiner and *b* is the Simple NF.

   (a) IF *a* is in the set of Determiner allomorphs, and *i* is EQUAL to 4 or 2, then

      i. IF the initial Xer of *b* is in the set of Lunar Xers: Lunax, then

         SET ans to FLEX *b*, and IF ans is not NULL, then

         IF *b* is Defective and ambiguous in DFN, then ASSIGN to *x* the property heritage 1: [the NGC & DFN: (M) S, npm, dfp], ASSIGN to *x* the FLXN & CAT of *b*, and DO aa = [ASSIGN to *x* the TYP of *a*, and RETURN *x*].

         IF *b* has no DFN value, then
         ◇ IF *b* is ambiguous in FLEXION, then PASSON2 to *x* the NGC, VAL, MRK, HTY, & CAT of *b*, and DO bb = [ASSIGN to *x* the FLXN & DFN: svm, dfp, and DO aa].
         ◇ IF *b* has a reduced NGC suffix, and *b* is a dual Numeral with a value 2, 200 or 2000, then PASSON2 to *x* the NGC, FLXN, VAL, MRK, & CAT of *b*, and DO cc = [ASSIGN to *x* the DFN: dfp, and DO aa].
         ◇ ELSE PASSON2 to *x* the NGC, FLXN, VAL, MRK, HTY, & CAT of *b*, and DO cc.
         ◇ ELSE FAIL.

   (b) ELSE SET *c* to the initial Xer of *b*, and SET *d* to the 2nd element of SEGM 1 of *b*, then

      i. IF *i* is EQUAL to 3 or 1, and IF *c* is in the set of Solar Xers: Solax, and *c* is EQUAL to the initial Xer of *d*, then

         SET ans to FLEX *d*, then

         IF *d* is Defective and ambiguous in DFN, then ASSIGN to *x* the property heritage 1, the FLXN & CAT of *d*, and DO aa.

         IF *d* has no DFN value, then
         ◇ IF *d* is ambiguous in FLEXION, then PASSON2 to *x* the NGC, VAL, MRK, HTY, & CAT of *d*, and DO bb.
         ◇ IF *d* has a reduced NGC suffix, and *d* is a dual Numeral with a value 2, 200 or 2000, then PASSON2 to *x* the NGC, FLXN, VAL, MRK, & CAT of *d*, and DO cc.
         ◇ ELSE PASSON2 to *x* the NGC, FLXN, VAL, MRK, HTY, & CAT of *d*, and DO cc.
         ◇ ELSE FAIL.

   (c) ELSE SET *i* to *i* MINUS 1, and GOTO *STEP I.* 1.

3./*STEP II./STEP C.* ELSE FAIL.

SCHEME 27: DEFINITIZE1: An Iterative Left-to-Right Scheme for the Recognition of DNFs

---

ARGUMENTS: an inputword x for CNF.
OBJECTIVES:
† recognize a Complex NF which is a Simple NF attached with an optional GP and which satisfies affixation constraints.
† apply compulsory simple transformations where appropriate.
LOCAL VARIABLES: i, a, b, c, d, and ss.
BASIC METHOD:
STEP A:
IF the final Xer of x is in the set of ANF boundaries: {a, i, o, u, w, y, |}, then %%% entry condition.
SET i to 2, then
STEP I:
IF i is GREATER than or EQUAL to 6, then FAIL.
STEP II:
IF i is strictly LESS than 6, then DO: SET ss to SEGMV i of x, SET a to the 1st and b to the 2nd elements of ss, then %%% a is the suffix and b the centre.

1. IF a is the GP 'ya', then SET c to the 1st element of SEGMV 2 of b, then

(a) IF either MVACT of b or TVACT2 of b is not NULL, then

i. IF b is a Noun or a Verbal1, then

IF c is the Defective suffix 'iy', then

IF b is ambiguous between a MRP and the singular, then DO aa = [PASSON2 to x the NG & HTY of b, ASSIGN to x the CMK, FLXN, DFN, & REF: ncp, svm, dfp, col, and RETURN a list of b and a].

IF b is a MRP, then ASSIGN to x these NGC, FLXN, HTY, DFN, & REF: (M) P, ncp, svm, hm, dfp, col, and RETURN a list of b and a.

IF b is a Defective and b is plural, then PASSON2 to x the NG, HTY, & TYP of b, ASSIGN to x the CMK, FLXN, DFN, & REF: ncp, svm, dfp, col, and RETURN a list of b and a.

ELSE FAIL.

IF b has a reduced NGC suffix, then

IF b is dual and nominative, then PASSON2 to x the NGC & FLXN of b, ASSIGN to x the DFN & REF: dfp, col, and RETURN a list of b and a.

/ii. ELSE FAIL.

(b) IF *c* is the Abbreviated suffix 'ay', SET *d* to MKWORD of *b* and <o>, then

i. IF either MVACT of *d* or TVACT2 of *d* is not NULL, then

IF *d* is dual, then

IF *d* is a Noun or a Verbal1, then DO bb = [PASSON2 to *x* the NGC of *d*, ASSIGN to *x* the FLXN, DFN, & REF: svm, dfp, col, and RETURN a list of *b* and *a*].

/ii. ELSE FAIL.

(c) IF *c* is the suffix 'a|', SET *d* to MKWORD of *b* after SUBSTITUTing <|> with <y>, then

i. IF FLEX *d* is not NULL, then

IF either *d* is a Noun or *d* is not a Numeral and has ambiguous FLEXION, then DO bb.

/ii./(d) ELSE FAIL.

2. IF *a* is the suffix 'iy', then

(a) IF either MVACT of *b* or TVACT2 of *b* is not NULL, then

i. IF *b* is not Defective, then

IF *b* is a Noun, a Verbal1 or a Verbal2, then DO aa.

/ii./(b) ELSE FAIL.

3. IF *a* is in the other GPs, then

(a) IF the initial and final Xers of *a* are not <y>, then

i. IF FLEX *b* is not NULL, then

IF *b* is ambiguous in DFN, then

IF *b* is Defective, then DO cc = [PASSON2 to *x* the NGC & HTY of *b*, ASSIGN to *x* the FLXN & DFN: svm, dfp, the REF of *a*, and RETURN a list of *b* and *a*].

ELSE PASSON2 to *x* the NGC of *b*, ASSIGN to *x* the FLXN, DFN, & HTY: svm, dfp, hm, the REF of *a*, and RETURN a list of *b* and *a*].

ELSE FAIL.

IF *b* does not have the FLEXION of a full NGC suffix and is devoid of DFN, then DO cc.

ELSE FAIL.

ii. IF the final two Xers of *b* are the suffix 'a|', then

IF the stem of *b* is in the set: {:b, :x}, then SET *d* to the 2nd element of SEGMV 1 of *b*, then DO dd = [
IF FLEX *d* is not NULL, then
◇ IF *d* is singular and masculine and is devoid of DFN, then DO ee = [PASSON2 to *x* the NGC of *d*, ASSIGN to *x* the FLXN & DFN: svm, dfp, the REF of *a*, and RETURN a list of *b* and *a*].
◇ ELSE FAIL].

ELSE SET *d* to MKWORD of *b* after SUBSTITUTing <|> with <y>, then

IF FLEX *d* is not NULL, then
◇ IF *d* is devoid of DFN, then DO ee.
◇ ELSE FAIL.

iii. IF the final Xer of *b* is either <w> or <y>, then

IF the stem of *b* is in the set: {:b, :x}, then SET *d* to the 2nd element of SEGMV 1 of *b*, then DO dd.

/iv./(b) ELSE FAIL.

4. ELSE increment *i* by 1 and GOTO *STEP II.*

STEP III./STEP B. ELSE FAIL.

SCHEME 28: DEFINITIZE2: An Iterative Right-to-Left Scheme for the Recognition of ANFs

---

ARGUMENTS: an inputword *x* for Complex NF.
OBJECTIVES:
monitor the output of DEFINITIZE1 and DEFINITIZE2, enforce the observation of the TRANSITIVITY and Graphotactic Conditions of NF$GP affixation, and perform categorial disambiguation on the output of DEFINITIZE1 and 2.
LOCAL VARIABLES: *a*, *b*, and ans.
BASIC METHOD:
STEP A:
SET ans to DEFINITIZE1 of *x*, and IF ans is not NULL, then
STEP I:
IF *x* is Definite, then

1. IF *x* is a Numeral, then ASSIGN to *x* the CAT: DM, and RETURN ans.

2. IF *x* is an Adjective or a Verbal2, then ASSIGN to *x* the CAT: DA, and RETURN ans.

3. ELSE ASSIGN to x the CAT: DN, and RETURN ans.

4. ELSE FAIL.

STEP II:
IF x is Indefinite or ambiguous in DFN, then

1. IF x has either a full NGC suffix , or a full vowel mark, then RETURN ans.

2. IF x is a Diptote of Type 1, then

   (a) IF x is an Adjective, then DO aa = [ASSIGN to x the FLXN & CAT: xvm, XA, and RETURN ans].

   (b) IF x is a Noun, then ASSIGN to x the FLXN & CAT: xvm, XN, and RETURN ans.

   (c) IF x is a Df.Ab. Nominal, then

      i. IF x has a NUMERICAL VALUE, then ASSIGN to x the FLXN & CAT: svm, MM, and RETURN ans.

      ii. IF x is an Adjective, then DO aa.

      iii. ELSE FAIL.

3. IF x is devoid of TRVY or x is monotransitive, ditransitive, xtransitive, ptransitive2 or mxtransitive, then

   (a) IF x has the CAT: WN or is a Numeral, then RETURN ans.

   (b) ELSE ASSIGN to x the CAT: MD, and RETURN ans.

   (c)/4. ELSE FAIL.

STEP III:
ELSE embed x in the error message "error7", and RETURN "error7".
STEP B:
SET ans to DEFINITIZE2 of x, and IF ans is not NULL, then SET a to the 1st and b the 2nd elements of ans, then %%% a is the NF and b is the GP.
STEP I:
IF a is devoid of TRVY or monotransitive, ditransitive, xtransitive, ptransitive2 or mxtransitive, then

1. IF x is collocutive, then DO bb = [ASSIGN to x the CAT: AN, and RETURN ans].

2. IF x is exlocutive, then

(a) IF b is singular feminine, then DO bb.

(b) IF the final Xer of a is in the set: {i, y}, then

i. IF the 2nd Xer of b is <i>, then DO bb.

ii. ELSE embed x in the error message "error3", and RETURN "error3".

(c) IF the final Xer of a is <o>, then

i. IF the penultimate Xer of a is <y>, then

IF the 2nd Xer of b is <i>, then DO bb.

ELSE embed x in the error message "error8", and RETURN "error8".

ELSE FAIL.

ii. IF the 2nd Xer of b is <u>, then DO bb.

iii. ELSE embed x in the error message "error9", and RETURN "error9".

(d) IF the final Xer of a is in the set: {a, u, w, I}, then

i. IF the 2nd Xer of b is <u>, then DO bb.

ii. ELSE embed x in the error message "error4", and RETURN "error4".

iii./(e)/3. ELSE embed x in the error message "error7", and RETURN "error7".

STEP II./STEP C. ELSE FAIL.

SCHEME 29: FILTER4: A Procedural Scheme for Graphotactic and TRANSITIVITY Filtering for CNFs

---

ARGUMENTS: an inputword x for Complex NF.
OBJECTIVES:
† recognize a Complex NF which is a composite structure of an optional Bound Preposition P and DNF or P and ANF.
† enforce oblique CASE GOVERNMENT conditions.
† perform CASE disambiguation.
LOCAL VARIABLES: a, b, s, res, and ans.
BASIC METHOD:
STEP A:
SET res to FILTER4 of x, and IF res is not NULL, then
STEP I:

IF res is an error, then RETURN res.
STEP II:
IF x is not bound, then RETURN res.
STEP III: ELSE FAIL.
STEP B:
ELSE IF the 1st Xer of x is in the set of PNF boundaries: {b, k, l}, then SET s to SEGM 2 of x, SET a to the 1st and b to the 2nd elements of s, then %%% a is the Preposition and b the centre.
STEP I:
IF a is the Preposition 'li', then

1. SET ans to FILTER4 of b, and IF ans is not NULL, then

   (a) IF ans is an error, then RETURN ans.

   (b) IF b is oblique, ambiguous in DFN or ambiguous in CMK, then

      i. IF b is Annexed, then DO aa = [PASSON2 to x the NG, FLXN, VAL, MRK, DFN, HTY, & REF of b, ASSIGN to x the CMK: obm, a CAT which is P attached to the CAT of b, and RETURN a list which is MKWORD of a and the 1st element of ans, and the 2nd element of ans]. %%% these are the NF and the GP.

      ii. ELSE IF b is bound, a Diptote of Type 1, ambiguous in CMK, Defective or devoid of TYP, then DO bb = [PASSON2 to x the NG, FLXN, VAL, MRK, DFN, HTY, TYP, & REF of b, ASSIGN to x the CMK: obm, a CAT which is P attached to the CAT of b, and RETURN x].

      iii. ELSE FAIL.

   (c) IF b is accusative, then

      i. IF b is a Diptote of Type 1, then

         IF b is bound and not Annexed, then DO bb.

         ELSE FAIL.

      ii. IF b is a bound singular Numeral and not Annexed, then DO cc = [PASSON2 to x the NG, FLXN, VAL, MRK, DFN, HTY, & REF of b, ASSIGN to x the CMK: ncp, and RETURN a CAT which is P attached to the CAT of b, and RETURN x].

      iii./(d)/2. ELSE FAIL.

STEP II:
IF a is in the Prepositions: {bi, ka}, then

1. SET ans to FILTER4 of b, and IF ans is not NULL, then,

   (a) IF ans is an error, then RETURN ans.

(b) IF *b* is oblique, ambiguous in DFN or ambiguous in CMK, then

i. IF *b* is Annexed, then DO aa.

ii. IF *b* is free, a Diptote of Type 1, ambiguous in CMK, Defective or devoid of TYP, then do bb.

iii. ELSE FAIL.

(c) IF *b* is accusative, then

i. IF *b* is a Diptote of Type 1, then

IF *b* is free and not Annexed, then DO bb.

ELSE FAIL.

ii. IF *b* is a free singular Numeral and not Annexed, then DO cc.

iii./(d)/2./*STEP III./STEP C.* ELSE FAIL.

SCHEME 30: GENITIVIZE2: A Procedural Left-to-Right Scheme for Prepositional CNF Recognition

---

ARGUMENTS: an inputword *x* for a Complex PF, or PGP.
OBJECTIVES:
† recognize a Complex PF that is an AP$CP sequence, or a PGP by performing Pronoun stripping and prefix identification based on dictionary look-up.
† impose Affix Selection Rules.
LOCAL VARIABLES: *i*, *a*, *b*, *c*, and *s*.
BASIC METHOD:
STEP A:
IF the initial and final Xers of *x* are in the set of CPF/PGP boundaries, SET *i* to 2, then
STEP I:
IF *i* is GREATER than, or EQUAL to 6, then FAIL.
STEP II:
IF *i* is strictly LESS than 6, then SET *s* to SEGMV *i* of *x*, SET *a* to the 1st & *b* the 2nd elements of *s*, then %%% *a* is the GP & *b* is the Pr.

1. IF *a* is EQUAL to 'iy', then SET *c* to MKWORD of *b* and <a>, then

(a) IF *c* is in the set of Free Prepositions, then PASSON to *x* the NG, DFN, & REF of *a*, ASSIGN to *x* the PRS: 1st, & DO aa = [ASSIGN to *x* the CMK & CAT: obm, PGP, & RETURN a list of *b* & *a*].

(b) IF *c* is in the set of Affirmative Particles, then PASSON to *x* the NGP, DFN, & REF of 'iy' & ASSIGN to 'iy' the CMK & CAT: acm, CP & to *x* the CMK & CAT:

acm, PC, & RETURN a list of *b* & *a*.

(c) IF the final Xer of *b* is <n>, then

i. IF the penultimate Xer of *b* is <n>, then
SET *c* to MKWORD of *b* after SUBSTITUTing <n> with <o>, then

IF *c* is in the set of Free Prepositions, then PASSON to *x* the NGP, DFN, & REF of 'niy', & DO aa.

ELSE FAIL.

ii. ELSE SET *c* to *b* with the final Xer DELETed, then

IF *c* is in the set of Affirmative Particles, then PASSON to *x* the NGP, DFN, & REF of 'niy' & ASSIGN to 'niy' the CMK & CAT: acm, CP & to *x* the CMK & CAT: acm, PC, & RETURN a list of *c* & 'niy'.

/iii./(d) ELSE FAIL.

2. IF *a* is EQUAL to 'ya', then

(a) IF *b* is in the set of Free Prepositions, then

i. IF the final Xer of *b* is <y>, then PASSON to *x* the NGP, DFN, & REF of 'ya', & DO aa.

ii./(b) ELSE FAIL.

3. IF *a* is EQUAL to 'na|', then

(a) IF the final Xer of *b* is <n>, then SET *c* to MKWORD of *b* & <o>, then

i. IF *c* is in the set of Free Prepositions, then DO bb = [PASSON to *x* the NGP, DFN, HTY, & REF of *a*, & DO aa].

ii. ELSE FAIL.

(b) IF *b* is in the set of Bound Prepositions: {la, bi}, then DO bb.

(c) IF *b* is in the set of Affirmative Particles, then DO cc = [PASSON to *x* the NGP, DFN, HTY, REF of *a*, ASSIGN to *a* the CMK & CAT: acm, CP & to *x* the CMK & CAT: acm, PC, & RETURN a list of *b* & *a*].

(d) IF *b* is in the set of Free Prepositions, then

i. IF the final Xer of *b* is <y>, then

IF the penultimate Xer of *b* is not <a>, then DO bb.

ELSE FAIL.

ii. IF the penultimate Xer of *b* is not <n>, then DO bb.

iii./(e) ELSE FAIL.

4. IF *a* is in the set of other GPs, then

(a) IF the initial & final Xers of *a* are not <y>, then

i. IF *b* is in the set of Bound Prepositions: {la, bi}, then DO bb.

ii. IF *b* is in the set of Affirmative Particles, then DO cc.

iii. IF *b* is in the set of Free Prepositions, then

IF the final Xer of *b* is <y>, then

IF the penultimate Xer of *b* is not <a>, then DO bb.

ELSE FAIL.

ELSE DO bb.

/iv./(b) ELSE FAIL.

5. ELSE SET *i* to *i* PLUS 1, and GOTO *STEP I.*

STEP III./STEP B. ELSE FAIL.

SCHEME 31: GENITIVIZE1: An Iterative Right-to-Left Scheme for the Recognition of CPFs and PGPs

---

ARGUMENTS: an inputword *x* for a Complex PF, or PGP.
OBJECTIVES:
† monitor and enforce the observation of Vocalic Compatibility conditions in the output structures from GENITIVIZE1.
† recognize Complex PF, or PGP sequences and enforce conditions of affixation.
LOCAL VARIABLES: *a*, *b*, *s*, & ans.
BASIC METHOD:
STEP A:
SET ans to GENITIVIZE1 of *x* & IF ans is not NULL, then SET *a* to the 2nd element in ans, then %%% *a* is the Pronoun.
STEP I:
IF *a* is collocutive, then RETURN ans.
STEP II:
IF *a* is exlocutive, then

1. IF *a* is the singular feminine, then RETURN ans.

2. IF the final Xer of the 1st element of ans is in the set: {i, y}, then
%%% this is the prefix.

   (a) IF the 2nd Xer of *a* is <i>, then RETURN ans.

   (b) ELSE embed *x* in the error message: "error3", and RETURN "error3".

3. IF the final Xer of the 1st element of ans is <o>, then

   (a) IF the penultimate Xer of this 1st element is <y>, then

      i. IF the 2nd Xer of *a* is <i>, then RETURN ans.

      ii. ELSE embed *x* in the error message: "error8", and RETURN "error8".

   (b) IF the 2nd Xer of *a* is <u>, then RETURN ans.

   (c) ELSE embed *x* in the error message: "error9", and RETURN "error9".

4. IF the final Xer of the 1st element of ans is <a>, then

   (a) IF the 2nd Xer of *a* is <u>, then RETURN ans.

   (b) ELSE embed *x* in the error message: "error4", and RETURN "error4".

STEP III:
IF the initial Xer of *x* is in the set of CPF, or PGP boundaries: {b, k, l}, then SET *s* to SEGM 2 of *x*, SET *a* to the 1st & *b* to the 2nd elements of ans, then

1. IF *a* is EQUAL to 'la', then

   (a) IF *b* is in the set of Future, or Assertive Particles: {sawofa, qado}, then ASSIGN to *x* the MDG of *b* & a CAT which is P attached to the CAT of *b* & RETURN a list of *x*.

   (b) ELSE FAIL.

2. IF *a* is in the set of Bound Prepositions: {li, bi, ka}, then

   (a) IF *b* is the Negative Particle 'la|' & *a* is 'bi', then ASSIGN to *x* the GVT: obm, a CAT value: PEP, & RETURN a list of *x*.

   (b) IF *a* is not 'ka' & *b* is <y>, then PASSON to *x* the NG, DFN, & REF of 'iy', ASSIGN to *x* the PRS, CMK, & CAT: 1st, obm, PGP, & RETURN a list of the 1st Xer of *a* & *b*.

(c) IF *b* is the Affirmative Particle: ':anna', then ASSIGN to *x* the GVT & CAT: acm, PP, & RETURN a list of *x*.

(d) IF *b* is in the set of Interrogative Particles: {mano, ma|, ma|d-a|}, then ASSIGN to *x* a CAT which is P attached to the CAT of *b* & the MODE & CMK: int, obm, & RETURN a list of *x*.

(e) ELSE SET ans to GENITIVIZE1 of *b*, then

i. IF the CAT of *b* is PC, then

IF the 2nd Xer of *b* is <a>, then PASSON to *x* the NGP, DFN, HTY, REF, & CMK of *b*; ASSIGN to *x* the CAT: PPC, & RETURN a list of *a* attached to the 1st element of ans, & the 2nd element of ans.

ELSE FAIL.

ii. ELSE RETURN ans.

(f)/3./*STEP IV./STEP B.* ELSE FAIL.

SCHEME 32: FILTER5: A Procedural Left-to-Right Scheme for the Processing and Graphotactic Filtering of CPFs and PGPs

---

ARGUMENTS: an inputword *w* for a Polysynthetic Word (PW).
OBJECTIVES:
recognize a PW through Interrogative and or Coordinative prefix identification, and dictionary look-up for Free Particles and Adverbs, or processing for other types of Complex Forms.
LOCAL VARIABLES: *a*, *b*, *c*, *d*, *s*, ss, res, and ans.
BASIC METHOD:
STEP A:
IF *w* is in the Function Word Table: FWT, then
IF *w* is not a bound allomorph, then RETURN *w*.
STEP B:
SET res to YPARSE *w* & IF res is not NULL, then RETURN res.
%%% YPARSE calls FILTER5, FUTURIZE, or GENITIVIZE2, in that order.
STEP C:
ELSE, IF the initial Xer of *w* is in the set of PW-Boundaries, SET *s* to SEGM 2 of *w*, SET *a* to the 1st & *b* to the 2nd elements of *s*, then
STEP I:
IF *a* is a Coordinative Conjunction, then

1. IF *b* is in FWT, then

(a) IF *b* is not a bound allomorph, then PASSON to *w* the GVT & MDG of *b*, & DO aa = [ASSIGN to *w* a CAT which is C attached to the CAT of *b* & RETURN a list of *w*].

(b) ELSE FAIL.

2. ELSE SET ans to YPARSE *b* & IF ans is not NULL, then

   (a) IF ans is an error, then RETURN ans.

   (b) IF *b* is collocutive or exlocutive, then PASSON2 the PLIST of *b* to *w*, ASSIGN to *w* a CAT which is C attached to the CAT of *b* & RETURN a list which is *a* attached to the 1st element of ans & the 2nd element of ans.

   (c) ELSE PASSON2 to *w* the PLIST of *b*, & DO aa.

   (d)/3. ELSE FAIL.

STEP II:
IF *a* is the Bound Interrogative Particle, then SET ss to SEGM 2 of *b*, SET *c* to the 1st & *d* to the 2nd elements of ss, then

1. IF *b* is in FWT, then

   (a) IF *b* is not a bound allomorph & *b* does not have the MODE: interrogative, then PASSON to *w* the GVT & MDG of *b*, & DO bb = [ASSIGN to *w* the MODE: int, & a CAT which is Q attached to the CAT of *b* & RETURN a list of *w*].

   (b) ELSE FAIL.

2. IF *c* is a Coordinative Conjunction, then

   (a) IF *d* is in FWT, then

      i. IF *d* is not a bound allomorph & *d* does not have the MODE: interrogative, then PASSON to *w* the GVT & MDG of *d*, & DO cc = [ASSIGN to *w* the MODE: int, & a CAT which is QC attached to the CAT of *d* & RETURN a list of *w*].

      ii. ELSE FAIL.

   (b) SET ans to YPARSE *d* & IF ans is not NULL, then

      i. IF ans is an error, then RETURN ans.

      ii. IF *d* is collocutive or exlocutive, then PASSON2 the PLIST of *d* to *w*, ASSIGN to *w* the MODE: int & a CAT which is QC attached to the CAT of *d* & RETURN a list which is *a* attached to *c* attached to the 1st element of ans & the 2nd element of ans.

      iii. ELSE PASSON2 to *w* the PLIST of *d*, & DO cc.

iv./(c) ELSE FAIL.

3. SET ans to YPARSE *b* & IF ans is not NULL, then

   (a) IF ans is an error, then RETURN ans.

   (b) IF *b* is collocutive or exlocutive, then PASSON2 the PLIST of *b* to *w*, ASSIGN to *w* the MODE: int, a CAT which is Q attached to the CAT of *b* & RETURN a list which is *a* attached to the 1st element of ans & the 2nd element of ans.

   (c) ELSE PASSON2 to *w* the PLIST of *b*, & DO bb.

   (d)/4./*STEP III./STEP D.* ELSE FAIL.

SCHEME 33: QUESTION: A Procedural Left-to-Right Scheme for PW-Recognition

ARGUMENTS: an inputword w for a Polysynthetic Word (PW).
OBJECTIVES:
† filter out idiosyncratically ungrammatical CATEGORIES.
† perform categorial disambiguation in PW-Contexts.
LOCAL VARIABLE: ans.
BASIC METHOD:
STEP A:
SET ans to QUESTION w & IF ans is not NULL, then
STEP I:
IF ans is an error, then RETURN ans.
STEP II:
IF the CAT of w is YXA where Y is P, Q, CP, QP, QC, or QCP, then MODIFY XA to MD & RETURN ans.
STEP III:
IF the CAT of w is YVM where Y is Q or QC then MODIFY VM to VB & RETURN ans.
STEP IV:
IF the CAT of w is YWV where Y is Q or QC, then ASSIGN to the Pronoun in WV a CAT2: CP, then MODIFY WV to VR & RETURN ans.
STEP V:
IF the CAT of w is QPP or QPPC, then

1. IF the 3rd Xer of w is not <b>, then RETURN ans.

2. ELSE FAIL.

STEP VI:
IF the CAT of w is QCPP or QCPPC, then

1. IF the 5th Xer of w is not <b>, then RETURN ans.

2. ELSE FAIL.

STEP VII:
IF the CAT of w is not in the set: {QPJQ, QCPJQ, QPHQ, QCPHQ, QPQP, QCPQP}, then RETURN ans.
STEP VIII: ELSE FAIL.

SCHEME 34: FILTER6: A Procedural Scheme for Categorial Filtering and Disambiguation

ARGUMENTS: a list L of inputwords, each being a Polysynthetic Word *w*.
OBJECTIVE:
ASSIGN a default value: third for the property PERSON, to forms that are devoid of values for PERSON. %%% WPARSE.
LOCAL VARIABLE: ans.
BASIC METHOD:
STEP A:
SET ans to FILTER6 *w* & IF ans is not NULL, then
STEP I:
IF ans is an error, then RETURN ans.
STEP II:
IF *w* is devoid of PERSON values, then ASSIGN to *w* the PRS: 3rd, & RETURN ans.
STEP III:
ELSE RETURN ans.
STEP B:
ELSE RETURN *w* embedded in the error message "!*error".
OBJECTIVES:
† recognize recursively the list of inputwords or sentence.
† distinguish referential outputwords in order to store copies of their clitic Pronouns in global registers. %%% MPARSE.
LOCAL VARIABLES: mreg, ans.
BASIC METHOD:
SET the global register rreg to NIL, then
STEP A:
IF the list L is empty, then RETURN the REVERSE of the LOCAL register mreg.
STEP B:
SET ans to WPARSE the 1st element of ans (*w*), & IF ans is not NULL, then
STEP I:
IF ans is an error, then RETURN ans.
STEP II:
IF the 1st element of L is collocutive or exlocutive then PUSH a COPY of the 2nd element of ans into rreg, PUSH ans into mreg, SET L to the REST of L, & GOTO *STEP A*.
STEP III:
ELSE PUSH ans into mreg, SET L to the REST of L, & GOTO *STEP A*.
STEP IV./STEP C. ELSE FAIL.

SCHEME 35: MPARSE: An Iterative Left-to-Right Procedure for the Recognition of Sentences

ARGUMENTS: x for the output of a morphological parse of a word.
OBJECTIVES:
† PAIR the word and its CATEGORY—for a non-referential form.
† PAIR the word and its CATEGORY, and PAIR the Pronoun and its CATEGORY—for a referential form. %%% EXSYMBOLS.
LOCAL VARIABLE: c.
BASIC METHOD:
STEP A:
IF x is an ATOM or NULL, then FAIL.
STEP B:
IF the REST of x is NULL, then RETURN a list of the CAT of x & x.
STEP C:
IF x is a list, then SET c to MKWORD of the 1st & 2nd elements of x, then RETURN a list from PAIRing the CAT of c & c, & PAIRing the CAT of the 2nd element of x & the 2nd element of x.
STEP D:
ELSE FAIL.
ARGUMENTS: L as a list of inputwords or sentence. %%% MKTRUNK.
OBJECTIVE:
MODIFY the output of MPARSE by constructing an output MORPHOLOGICAL TRUNK, or primitive tree—consisting of CATEGORIES and terminal SYMBOLS—to serve as input for syntactic parsing.
LOCAL VARIABLES: ans, trunk.
BASIC METHOD:
RESET, then %%% this is a function which resets the PLISTS.
STEP A:
SET ans to MPARSE L, & IF ans is not NULL, then
STEP I:
IF ans is an error, then RETURN ans.
STEP II:
ELSE DO:

1. IF the 1st element of ans is NULL, then RETURN the REVERSE of trunk.

2. ELSE PUSH the result of EXSYMBOLS of the 1st element of ans into trunk, then SET ans to the REST of ans, & GOTO STEP 1.

3./*STEP III./STEP B.* ELSE FAIL.

SCHEME 36: MKTRUNK: An Iterative Left-to-Right Procedure for the Generation of Output M-Trees

==0==<★★★★★>==0==

Appendix D
PROGRAM TRACES

Script started on Thu Apr 26 11:37:36 1990;
xenakis %%% LISP
Cambridge Orion1.05/UNIX LISP System entered in about 1,550 Kbytes.
Store image was made at 21:08:20; on 12 Apr 1990;
LISP version—1.10/1.10/639950892; image size = 92,340 bytes.

Input: (fload TUNIS1)
Value: nil

A. SAMPLES OF OUTPUT AND TIMING FROM TUNIS1

%%% single words

Input: (tm '(exroot 'kataba rootlist))
(time taken 2.0e−2 SECS)
%%% (e-2: multiplied by 10 to the power of -2; e.g., 2.0e−2 = 0.002)
Value: ktb

Input: (tm '(derive3 'kawwan !*patlist))
(time taken 2.0e−2 SECS)
Value: (patlist1 kwwn cawwac)

Input: (tm '(derive3 'ajolis patlist+))
(time taken 2.0e−2 SECS)
Value: (patlist3 jls acocic)

Input: (tm '(mkperfect 'kunotu))
(time taken 1.0e−1 SECS)
Value: ((kun otu) patlist1 kn cuc)

Input: (tm '(mkperfect 'banawo!))
(time taken 1.6e−1 SECS)
Value: ((ban awo!) patlist1 bn cac)

Input: (tm '(mkimperfect 'taquwmu))
(time taken 2.0e−2 SECS)
Value: ((t aquwm u) patlist3 qm acuwc)

Input: (tm '(mkimperfect 'tugannawona))
(time taken 3.2e−1 SECS)
Value: ((t ugann awona) patlist3 gnn ucacc)

Input: (tm '(filter1 'bitotu))
(time taken 1.2e−1 SECS)
Value: "error2: Verb-spelling violation: c_1oc_1! in: bitotu"

Input: (tm '(filter1 'katabna))
(time taken 1.0e−1 SECS)
Value: "error1: Verb-spelling violation: c_1c_2 or ec! in: katabna"

Input: (tm '(filter2 'tah-osunona))
(time taken 2.2e−1 SECS)
Value: "error2: Verb-spelling violation: c_1oc_1! in: tah-osunona"

Input: (tm '(filter2 'taqotulna))
(time taken 1.2e−1 SECS)
Value: "error1: Verb-spelling violation: c_1c_2 or ec! in: taqotulna"

Input: (tm '(tvact1 'h-asiba))
(time taken 6.0e−2 SECS)
Value: (h-asib a)

Input: (tm '(tvact1 'qutila))
(time taken 6.0e−2 SECS)
Value: (qutil a)

Input: (tm '(xparse 'kassaruwhu))
(time taken 5.0e−1 SECS)
Value: ((kassaruw) (hu))

Input: (tm '(xparse 'qalamahu))
(time taken 2.2e−1 SECS)
Value: ((qalama) (hu))

Input: (tm '(filter3 'kutibahu))
(time taken 2.4e−1 SECS)
Value: "error6: illegal morph-combination: VB-CP; TRANSITIVITY violation in: kutibahu"

Input: (tm '(filter3 'banayotuhi))
(time taken 4.4e−1 SECS)
Value: "error4: incompatible vowel-sequence: u/a/w/| + v $\neq$ u! in: banayotuhi"

Input: (tm '(futurize 'sayarokabuhu))
(time taken 1.92 SECS)
Value: ((sayarokabu) (hu))

Input: (tm '(futurize 'litah-omilahu))
(time taken 2.1 SECS)
Value: ((litah-omila) (hu))

Input: (tm '(mktrunk '(satabonuwnahu)))
(time taken 2.56 SECS) ("used in Table 63")
Value: ((((PVR . satabonuwnahu) (CP . hu))) ((satabonuwnahu pl mc r2 Ø Ø Ø Ø Ø Ø Ø Ø PVR Ø imp Ø act mtr ind Ø Ø exl Ø) (hu sn mc r3 Ø dfp Ø Ø acm Ø Ø Ø Ø CP Ø Ø Ø Ø Ø Ø Ø exl Ø)))

Input: (tm '(mktrunk '(sayad;oribuwnahunna)))
(time taken 3.52 SECS) ("used in Table 63")
Value: ((((PVR . sayad;oribuwnahunna) (CP . hunna))) ((sayad;oribuwnahunna pl mc r3 Ø Ø Ø Ø Ø Ø Ø Ø PVR Ø imp Ø act mtr ind Ø Ø exl Ø) (hunna pl ff r3 hm dfp Ø nmm acm svm Ø Ø Ø CP Ø Ø Ø Ø Ø Ø Ø exl Ø)))

Input: (tm '(mktrunk '(qatalotuma|hu)))
(time taken 8.8e−1 SECS)
Value: ((((VR . qatalotuma|hu) (CP . hu))) ((qatalotuma|hu dl mf r2 Ø Ø Ø Ø Ø Ø Ø Ø VR Ø prf Ø act mtr Ø Ø Ø exl Ø) (hu sn mc r3 Ø dfp Ø Ø acm Ø Ø Ø Ø CP Ø Ø Ø Ø Ø Ø Ø exl Ø)))

Input: (tm '(mktrunk '(yukawwinuwna)))
(time taken 6.6e−1 SECS)
Value: (((VB . yukawwinuwna)) ((yukawwinuwna pl mc r3 Ø Ø Ø Ø Ø Ø Ø Ø VB Ø imp Ø act mtr ind Ø Ø Ø Ø)))

Input: (tm '(mktrunk '(tamonah-onaniy)))
(time taken 1.56 SECS)
Value: ((((VR . tamonah-onaniy) (CP . niy))) ((tamonah-onaniy pl ff r2 Ø Ø Ø Ø Ø Ø Ø Ø VR Ø imp Ø act dtr jis Ø Ø col Ø) (niy sn mf r1 Ø dfp Ø Ø acm Ø Ø Ø Ø CP Ø Ø Ø Ø Ø Ø Ø col Ø)))

Input: (tm '(mktrunk '(tukattiba|nihima|)))
(time taken 1.9 SECS)
Value: ((((VR . tukattiba|nihima|) (CP . hima|))) ((tukattiba|nihima| dl mf rx Ø Ø Ø Ø Ø Ø Ø Ø VR Ø imp Ø act dtr ind Ø Ø exl Ø) (hima| dl mf r3 Ø dfp Ø Ø acm Ø Ø Ø Ø CP Ø Ø Ø Ø Ø Ø Ø exl Ø)))

Input: (tm '(mktrunk '(qatalawo|)))
(time taken 8.0e−1 SECS)
Value: "note : unknown word!: qatalawo|"

Input: (tm '(mktrunk '(taqumo)))
(time taken 2.4e−1 SECS)
Value: (((VB . taqumo)) ((taqumo sn mf rx Ø Ø Ø Ø Ø Ø Ø Ø VB Ø imp Ø act ntr jus Ø Ø Ø Ø)))

Input: (tm '(mktrunk '(takuwna)))
(time taken 2.6e−1 SECS)
Value: (((VB . takuwna)) ((takuwna sn mf rx Ø Ø Ø Ø Ø Ø Ø Ø VB Ø imp Ø act ctr sub Ø Ø Ø Ø)))

Input: (tm '(mktrunk '(:axod;aru)))
(time taken 3.0e−1 SECS)

Value: (((VA . :axod;aru)) ((:axod;aru sn mf r1 Ø ndf Ø nmm Ø svm Ø Ø VA Ø imp Ø act ntr ind Ø Ø Ø Ø)))

Input: (tm '(mktrunk '(d-ahaba)))
(time taken 6.0e−2 SECS) ("**used in Table 63**")
Value: (((VN . d-ahaba)) ((d-ahaba sn mc r3 Ø ndf Ø acm Ø svm Ø Ø VN Ø prf Ø act ptr1 Ø Ø Ø Ø Ø)))

Input: (tm '(mktrunk '(ɛasvara)))
(time taken 6.0e−2 SECS)
Value: (((VM . ɛasvara)) ((ɛasvara sn mc r3 Ø ndf Ø acm Ø svm 10 Ø VM Ø prf Ø act mtr Ø Ø Ø Ø Ø)))

Input: (tm '(mktrunk '(xa|laniy)))
(time taken 1.8e−1 SECS) ("**used in Table 63**")
Value: ((((VR . xa|laniy) (CP . niy))) ((xa|laniy sn mc r3 Ø Ø Ø Ø Ø Ø Ø Ø VR Ø prf Ø act mtr Ø Ø Ø col Ø) (niy sn mf r1 Ø dfp Ø Ø acm Ø Ø Ø Ø CP Ø Ø Ø Ø Ø Ø Ø col Ø)))

Input: (tm '(mktrunk '(qalamahu)))
(time taken 2.2e−1 SECS)
Value: ((((WV . qalamahu) (XP . hu))) ((qalamahu sn mc r3 Ø dfp Ø acm Ø Ø Ø Ø WV Ø prf Ø act mtr Ø Ø Ø exl Ø) (hu sn mc r3 Ø dfp Ø Ø cpm Ø Ø Ø Ø XP Ø Ø Ø Ø Ø Ø Ø exl Ø)))

Input: (tm '(mktrunk '(d-ahababu)))
(time taken 2.4e−1 SECS)
Value: ((((AN . d-ahababu) (GP . hu))) ((d-ahababu sn mc r3 Ø dfp Ø acm Ø Ø Ø Ø AN Ø Ø Ø Ø Ø Ø Ø Ø exl Ø) (hu sn mc r3 Ø dfp Ø Ø obm Ø Ø Ø Ø GP Ø Ø Ø Ø Ø Ø Ø exl Ø)))

Input: (tm '(mktrunk '(sa:axod;aru)))
(time taken 6.8e−1 SECS)
Value: (((PVB . sa:axod;aru)) ((sa:axod;aru sn mf r1 Ø Ø Ø Ø Ø Ø Ø Ø PVB Ø imp Ø act ntr ind Ø Ø Ø Ø)))

Input: (tm '(mktrunk '(li:axod;ara)))
(time taken 7.2e−1 SECS)
Value: (((PVB . li:axod;ara)) ((li:axod;ara sn mf r1 Ø Ø Ø Ø Ø Ø Ø Ø PVB Ø imp Ø act ntr sub Ø Ø Ø Ø)))

Input: (tm '(exstem ':as;odiqa|: rootlist))
(time taken 2.0e−2 SECS)
Value: s;dq

Input: (tm '(exstem ':amokinat rootlist))
(time taken 2.0e−2 SECS)
Value: kn

Input: (tm '(derivep 'mudarris mlist+))
(time taken 2.0e−2 SECS)
Value: (dplist1 drrs mucaccic)

Input: (tm '(derivec 'kita|b mlist))

(time taken 2.0e−2 SECS)
Value: (nnlist2 ktb cica|c)

Input: (tm '(deriveq 'mugannay xlist3))
(time taken 2.0e−2 SECS)
Value: (dplist6 gnn mucaccay)

Input: (tm '(derivez 'kurama|: mlist))
(time taken 0.000000000000001 SECS)
Value: (adlist2 krm cucaca|:)

Input: (tm '(mkmasc1 'mudarris))
(time taken 2.0e−2 SECS)
Value: ((mudarris) dplist1 drrs mucaccic)

Input: (tm '(mkmasc2 't-ama|niy))
(time taken 6.0e−2 SECS)
Value: ((t-ama|niy) mmlist2 t-mn caca|c)

Input: (tm '(feminize1 'ba|niyat))
(time taken 6.0e−2 SECS)
Value: ((ba|niyat) dplist3 bn ca|ciy)

Input: (tm '(feminize2 'fara|svat))
(time taken 1.0e−1 SECS)
Value: ((fara|svat) nnlistc frsv caca|c)

Input: (tm '(dualize1 'mudarrisatayoni))
(time taken 1.6e−1 SECS)
Value: ((mudarrisatayoni) dplist1 drrs mucaccic)

Input: (tm '(dualize2 ':axawayoni))
(time taken 6.0e−2 SECS)
Value: ((:axawayoni) nnlist2 :x cac)

Input: (tm '(mpluralize1 'mubannawona))
(time taken 6.0e−2 SECS)
Value: ((mubannawona) dplist5 bnn mucaccay)

Input: (tm '(mpluralize2 'banuwna))
(time taken 4.0e−2 SECS)
Value: ((banuwna) nnlist3 bn cic)

Input: (tm '(mpluralize2 'ɛisvoruwna))
(time taken 6.0e−2 SECS)
Value: ((ɛisvoruwna) mmlist2 ɛsvr cacoc)

Input: (tm '(fpluralize1 'mut-annaya|t))
(time taken 2.0e−2 SECS)
Value: ((mut-annaya|t) dplist6 t-nn mucaccay)

Input: (tm '(fpluralize2 ':axawa|t))
(time taken 4.0e−2 SECS)
Value: ((:axawa|t) nnlist3 :x cuc)

Input: (tm '(fpluralize2 'wah-ada|t))
(time taken 1.0e−1 SECS)
Value: ((wah-ada|t) nnlist3 h-d wicoc)

Input: (tm '(bpluralize 'svawa|qiy))
(time taken 1.0e−1 SECS)
Value: ((svawa|qiy) dplist3 svq cawa|c)

Input: (tm '(bpluralize 'nawa|si+))
(time taken 8.0e−2 SECS)
Value: ((nawa|si+) dplist3 ns cawa|c)

Input: (tm '(bpluralize 'mana|zil))
(time taken 8.0e−2 SECS)
Value: ((mana|zil) cvlist1 nzl maca|cic)

Input: (tm '(mvact 'wa|h-id))
(time taken 2.0e−2 SECS)
Value: ((wa|h-id) mmlist1 h-d wa|cic)

Input: (tm '(mvact 'rajul))
(time taken 0.000000000000001 SECS)
Value: ((rajul) nnlist2 rjl cacuc)

Input: (tm '(tvact2 'h-a|sib))
(time taken 4.0e−2 SECS)
Value: ((h-a|sib) dplist2 h-sb ca|cic)

Input: (tm '(tvact2 'mah-osuwb))
(time taken 4.0e−2 SECS)
Value: ((mah-osuwb) dplist4 h-sb macocuwc)

Input: (tm '(flex 'mudarrisa|ti+))
(time taken 7.6e−1 SECS)
Value: (mudarrisa|ti+)

Input: (tm '(flex 'ɛawa|t;ifu+))
(time taken 2.4e−1 SECS)
Value: nil

Input: (tm '(definitize1 ':alomi:ata|))
(time taken 1.02 SECS)
Value: (:alomi:ata|)

Input: (tm '(definitize1 ':alrrajulu))
(time taken 6.4e−1 SECS)
Value: (:alrrajulu)

Input: (tm '(definitize2 'mudarrisiyya))
(time taken 3.6e−1 SECS)
Value: ((mudarrisiy) (ya))

Input: (tm '(definitize2 'mabona|hu))
(time taken 5.600000000000001e−1 SECS)
Value: ((mabona|) (hu))

Input: (tm '(definitize2 'kita|bayya))
(time taken 4.0e−1 SECS)
Value: ((kita|bay) (ya))

Input: (tm '(filter4 'madoruwsu+))
(time taken 5.2e−1 SECS)
Value: (madoruwsu+)

Input: (tm '(filter4 'madoruwsu))
(time taken 5.0e−1 SECS)
Value: "error7: illegal morph-combination: VL-CP; TRANSITIVITY violation in: madoruwsu"

Input: (tm '(filter4 'kita|bihu))
(time taken 7.0e−1 SECS)
Value: "error3: incompatible vowel-sequence: i/y + v ≠ i! in: kita|bihu"

Input: (tm '(filter4 'ja|lisuhu))
(time taken 7.8e−1 SECS)
Value: "error7: illegal morph-combination: VL-CP; TRANSITIVITY violation in: ja|lisuhu"

Input: (tm '(genitivize2 'lilrrajuli))
(time taken 1.16 SECS)
Value: (lilrrajuli)

Input: (tm '(genitivize2 'likita|buhu))
(time taken 1.62 SECS)
Value: nil

Input: (tm '(mktrunk '(qa|tila|ta+)))
(time taken 6.4e−1 SECS)
Value: "note : unknown word!: qa|tila|ta+"

Input: (tm '(mktrunk '(kariymu+)))
(time taken 2.6e−1 SECS) ("used in Table 63")
Value: (((AA . kariymu+)) ((kariymu+ sn mc r3 Ø ndf Ø nmm Ø fvm Ø Ø AA Ø Ø Ø Ø
Ø Ø Ø Ø Ø Ø)))

Input: (tm '(mktrunk '(:it-ona|ni)))
(time taken 9.0e−1 SECS)
Value: (((MM . :it-ona|ni)) ((:it-ona|ni dl mc r3 Ø ndf Ø nmm Ø nnf 2 Ø MM Ø Ø Ø Ø
Ø Ø Ø Ø Ø Ø)))

Input: (tm '(mktrunk '(:abuwhu))))
(time taken 1.56 SECS)
Value: ((((AN . :abuwhu) (GP . hu))) ((:abuwhu sn mc r3 Ø dfp Ø nmm Ø svm Ø Ø AN Ø Ø Ø Ø Ø Ø Ø Ø Ø exl Ø) (hu sn mc r3 Ø dfp Ø Ø obm Ø Ø Ø Ø GP Ø Ø Ø Ø Ø Ø Ø exl Ø)))

Input: (tm '(mktrunk '(:abawayohi)))
(time taken 1.86 SECS)
Value: ((((AN . :abawayohi) (GP . hi))) ((:abawayohi dl mc r3 Ø dfp Ø cpm Ø svm Ø Ø AN Ø Ø Ø Ø Ø Ø Ø Ø Ø exl Ø) (hi sn mc r3 Ø dfp Ø Ø obm Ø Ø Ø Ø GP Ø Ø Ø Ø Ø Ø Ø exl Ø)))

Input: (tm '(mktrunk '(:ana|)))
(time taken 9.0e−1 SECS)
Value: (((DN . :ana|)) ((:ana| sn mf r1 Ø dfp Ø nmm Ø svm Ø Ø DN Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø)))

Input: (tm '(mktrunk '(layolay)))
(time taken 2.2e−1 SECS) ("used in Table 63")
Value: (((DN . layolay)) ((layolay sn ff r3 Ø dfp Ø ncp Ø svm Ø Ø DN Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø)))

Input: (tm '(mktrunk '(ba|niya+)))
(time taken 4.0e−1 SECS) ("used in Table 63")
Value: (((L1 . ba|niya+)) ((ba|niya+ sn mc r3 hp ndf Ø acm Ø fvm Ø Ø L1 Ø imp Ø act mtr Ø Ø Ø Ø Ø)))

Input: (tm '(mktrunk '(ba|niyuwna)))
(time taken 8.4e−1 SECS)
Value: "note : unknown word!: ba|niyuwna"

Input: (tm '(mktrunk '(:aloba|niy)))
(time taken 2.16 SECS)
Value: (((DN . :aloba|niy)) ((:aloba|niy sn mc r3 Ø dfp Ø npm Ø svm Ø Ø DN Ø Ø Ø Ø Ø Ø nbd Ø Ø Ø)))

Input: (tm '(mktrunk '(:alomukattibiy)))
(time taken 4.0 SECS)
Value: "note : unknown word!: :alomukattibiy"

Input: (tm '(mktrunk '(:alojabaluhu)))
(time taken 5.4 SECS) ("used in Table 63")
Value: "note : unknown word!: :alojabaluhu"

Input: (tm '(mktrunk '(:alorajulu)))
(time taken 1.86 SECS)
Value: "note : unknown word!: :alorajulu"

Input: (tm '(mktrunk '(:aloqalamu)))
(time taken 1.28 SECS)
Value: (((DN . :aloqalamu)) ((:aloqalamu sn mc r3 Ø dfp Ø nmm Ø svm Ø Ø DN Ø Ø Ø Ø Ø Ø nbd Ø Ø Ø)))

Input: (tm '(genitivize1 'baɛodiy'))
(time taken 0.000000000000001 SECS)
Value: ((baɛod) (iy))

Input: (tm '(genitivize1 ':innaniy))
(time taken 2.0e−2 SECS)
Value: ((:inna) (niy))

Input: (tm '(genitivize1 'fiyya))
(time taken 2.0e−2 SECS)
Value: ((fiy) (ya))

Input: (tm '(genitivize1 'minna|))
(time taken 2.0e−2 SECS)
Value: ((min) (na|))

Input: (tm '(genitivize1 'lana|))
(time taken 2.0e−2 SECS)
Value: ((la) (na|))

Input: (tm '(genitivize1 'lahumo))
(time taken 4.0e−2 SECS)
Value: ((la) (humo))

Input: (tm '(genitivize1 'maɛiy))
(time taken 0.000000000000001 SECS)
Value: ((maɛ) (iy))

Input: (tm '(filter5 ':ilayohuma|))
(time taken 2.0e−2 SECS)
Value: "error8: incompatible vowel-sequence: yo + v ≠ i! in: :ilayohuma|"

Input: (tm '(filter5 'lasawofa))
(time taken 4.0e−2 SECS)
Value: (lasawofa)

Input: (tm '(filter5 'liy))
(time taken 2.0e−2 SECS)
Value: ((l) (iy))

Input: (tm '(filter5 'li:annahunna))
(time taken 8.0e−2 SECS)
Value: ((li:anna) (hunna))

Input: (tm '(question ':afaka|tibu+))
(time taken 8.8e−1 SECS)
Value: (:afaka|tibu+)

Input: (tm '(question ':awahuna|))
(time taken 1.94 SECS)
Value: (:awahuna|)

Input: (tm '(filter6 ':awahalo))
(time taken 8.4e−1 SECS)
Value: nil

Input: (tm '(filter6 ':ah-asabahu))
(time taken 3.34 SECS)
Value: ((:ah-asaba) (hu))

Input: (tm '(filter6 ':aɛasvara))
(time taken 1.2 SECS)
Value: (:aɛasvara)

Input: (tm '(filter6 ':a:uwlay))
(time taken 1.28 SECS)
Value: (:a:uwlay)

Input: (tm '(mktrunk '(:afasatad;oribuhumo)))
(time taken 7.32 SECS) ("used in Table 63")
Value: ((((QCPVR . :afasatad;oribuhumo) (CP . humo))) ((:afasatad;oribuhumo sn mf rx Ø Ø Ø Ø Ø Ø Ø Ø QCPVR Ø imp Ø act mtr ind Ø Ø exl int) (humo pl mc r3 hm dfp Ø nmm acm svm Ø Ø Ø CP Ø Ø Ø Ø Ø Ø Ø exl Ø)))

Input: (tm '(mktrunk '(:awabimada|risihinna)))
(time taken 8.44 SECS) ("used in Table 63")
Value: ((((QCPAN . :awabimada|risihinna) (GP . hinna))) ((:awabimada|risihinna pl mf r3 Ø dfp Ø obm Ø svm Ø Ø QCPAN Ø Ø Ø Ø Ø Ø Ø Ø exl int) (hinna pl ff r3 Ø dfp Ø Ø obm Ø Ø Ø Ø GP Ø Ø Ø Ø Ø Ø Ø exl Ø)))

Input: (tm '(mktrunk '(:awabi:alomana|zili)))
(time taken 4.26 SECS)
Value: (((QCPDN . :awabi:alomana|zili)) ((:awabi:alomana|zili pl mf r3 Ø dfp Ø obm Ø svm Ø Ø QCPDN Ø Ø Ø Ø Ø Ø nbd Ø Ø int)))

%%% sentences

Input: (tm '(mparse '(:ana| rajulu+ saɛiydu+)))
(time taken 1.36 SECS)
Value: ((:ana|) (rajulu+) (saɛiydu+))

Input: (tm '(mparse '(had-ihi h-adiyqatu+ kabiyratu+ jamiylatu+)))
(time taken 1.78 SECS)
Value: ((had-ihi) (h-adiyqatu+) (kabiyratu+) (jamiylatu+))

Input: (tm '(mparse '(darasa :alowaladu :ald-d-akiyyu :alokita|ba :alojadiyda)))
(time taken 5.94 SECS)
Value: ((darasa) (:alowaladu) (:ald-d-akiyyu) (:alokita|ba)
(:alojadiyda))

Input: (tm '(mktrunk '(:anotuma| binota|ni jamiylata|ni)))
(time taken 1.86 SECS)

Value: ((DN . :anotuma|) (NN . binota|ni) (AA . jamiylata|ni))

Input: (tm '(mktrunk '(d;araba zayodu+ :alowalada :alsvsvaqiyya)))
(time taken 3.38 SECS) ("**used in Table 63**")
Value: ((VB . d;araba) (DN . zayodu+) (DN . :alowalada) (DA . :alsvsvaqiyya))

Input: (tm '(mktrunk '(kattaba samiyru+ :alobinota :alddarosa)))
(time taken 3.14 SECS)
Value: ((VB . kattaba) (DN . samiyru+) (DN . :alobinota) (DN . :alddarosa))

Input: (tm '(mktrunk '(d/anna :alrrajulu :alt;t;awiylu :alowasiymu :alomaro:ata :alojamiylata svaqiyyata+)))
(time taken 8.6 SECS)
Value: ((VB . d/anna) (DN . :alrrajulu) (DA . :alt;t;awiylu) (DA . :alowasiymu) (DN . :alomaro:ata) (DA . :alojamiylata) (AA . svaqiyyata+))

Input: (tm '(mktrunk '(:alobinotu :als;s;agiyratu :ald-d-akiyyatu h-a|sibatu+ :alrrajula :alt;t;awiyla ɛammaha|)))
(time taken 9.44 SECS) ("**used in Table 63**")
Value: ((DN . :alobinotu) (DA . :als;s;agiyratu) (DA . :ald-d-akiyyatu) (L1 . h-a|sibatu+) (DN . :alrrajula) (DA . :alt;t;awiyla) ((AN . ɛammaha|) (GP . ha|)))

Input: (tm '(mktrunk '(ɛalima :alowaladu :alofaqiyru :anna :alomudarrisata :alojadiydata satudarrisuhu :alokita|bata)))
(time taken 11.1 SECS) ("**used in Table 63**")
Value: ((VB . ɛalima) (DN . :alowaladu) (DA . :alofaqiyru) (AP . :anna) (DN . :alomudarrisata) (DA . :alojadiydata) ((PVR . satudarrisuhu) (CP . hu)) (DN . :alokita|bata))

Input: (tm '(mktrunk '(ɛallamato :alomudarrisatu :aloqadiymatu :alo:awola|da :alofuqara|:a :anna :alokita|ba t-amiynu+)))
(time taken 9.699999999999999 SECS) ("**used in Table 63**")
Value: ((VB . ɛallamato) (DN . :alomudarrisatu) (DA . :aloqadiymatu)
(DN . :alo:awola|da) (DA . :alofuqara|:a) (AP . :anna) (DN . :alokita|ba) (AA . t-amiynu+))

Input: (tm '(mktrunk '(la| rajula qas;iyra fiy :alomadorasati :alojamiylati)))
(time taken 4.82 SECS)
Value: ((EP . la|) (MD . rajula) (MD . qas;iyra) (Pr . fiy) (DN . :alomadorasati) (DA . :alojamiylati))

Input: (tm '(mktrunk '(fiy :alomanozili :als;s;agiyri :imora:atu+ kariymatu+)))
(time taken 4.22 SECS) ("**used in Table 63**")
Value: ((Pr . fiy) (DN . :alomanozili) (DA . :als;s;agiyri) (NN . :imora:atu+) (AA . kariymatu+))

Input: (tm '(mktrunk '(ɛinodiy qalamu+ t-amiynu+)))
(time taken 5.2e−1 SECS)
Value: (((PGP . ɛinodiy) (GP . iy)) (NN . qalamu+) (AA . t-amiynu+))

Input: (tm '(mktrunk '(layosa fiy :alobayoti rajulu+)))
(time taken 1.58 SECS)
Value: ((VB . layosa) (Pr . fiy) (DN . :alobayoti) (NN . rajulu+))

Input: (tm '(mktrunk '(ka|na zayodu+ fiy :alobayoti)))
(time taken 1.48 SECS)
Value: ((VB . ka|na) (DN . zayodu+) (Pr . fiy) (DN . :alobayoti))

Input: (tm '(mktrunk '(s;a|ra ɛamoru+ ganiyya+)))
(time taken 4.4e−1 SECS) ("used in Table 63")
Value: ((VB . s;a|ra) (DN . ɛamoru+) (AA . ganiyya+))

Input: (tm '(mktrunk '(zayodu+ huna|)))
(time taken 2.0e−1 SECS) ("used in Table 63")
Value: ((DN . zayodu+) (LD . huna|))

Input: (tm '(mktrunk '(:alddarosu makotuwbu+)))
(time taken 1.72 SECS)
Value: ((DN . :alddarosu) (L2 . makotuwbu+))

Input: (tm '(mktrunk '(:alobana|tu ka|tiba|tu+ :alddarosa)))
(time taken 3.58 SECS) ("used in Table 63")
Value: ((DN . :alobana|tu) (L1 . ka|tiba|tu+) (DN . :alddarosa))

Input: (tm '(mktrunk '(sawofa yamonah-u :alrrajulu :alokariymu :alobinota qalama+)))
(time taken 4.64 SECS)
Value: ((FD . sawofa) (VB . yamonah-u) (DN . :alrrajulu) (DA . :alokariymu) (DN . :alobinota)
(NN . qalama+))

Input: (tm '(mktrunk '(lano yad-ohaba :alowaladu :alsvsvaqiyyu :ilay :alomadorasati)))
(time taken 5.52 SECS) ("used in Table 63")
Value: ((SP . lano) (VB . yad-ohaba) (DN . :alowaladu) (DA . :alsvsvaqiyyu) (Pr . :ilay)
(DN . :alomadorasati))

Input: (tm '(mktrunk '(:alowaladu ka|tibu+ :alokita|ba)))
(time taken 3.08 SECS)
Value: ((DN . :alowaladu) (L1 . ka|tibu+) (DN . :alokita|ba))

Input: (tm '(mktrunk '(zayodu+ d;arabahu :alo:a|na ɛamoru+)))
(time taken 6.2e−1 SECS)
Value: ((DN . zayodu+) ((VR . d;arabahu) (CP . hu)) (AD . :alo:a|na) (DN . ɛamoru+))

Input: (tm '(mktrunk '(zayodu+ lamo yad;oribo ɛamora+)))
(time taken 7.4e−1 SECS)
Value: ((DN . zayodu+) (JP . lamo) (VB . yad;oribo) (DN . ɛamora+))

Input: (tm '(mktrunk '(:inna :alowalada d;arabahu :alomudarrisu)))
(time taken 3.44 SECS)
Value: ((AP . :inna) (DN . :alowalada) ((VR . d;arabahu) (CP . hu)) (DN . :alomudarrisu))

Input: (tm '(mktrunk '(:inna ɛamora+ d;araba zayoda+)))
(time taken 4.2e−1 SECS) ("used in Table 63")
Value: ((AP . :inna) (DN . ɛamora+) (VB . d;araba) (DN . zayoda+))

Input: (tm '(mktrunk '(kayofa :anota)))

(time taken 3.4e−1 SECS)
Value: ((QP . kayofa) (DN . :anɔta))

Input: (tm '(mktrunk '(kayofa qatala :alrrajulu samiyra+)))
(time taken 1.58 SECS)
Value: ((QP . kayofa) (VB . qatala) (DN . :alrrajulu) (DN . samiyra+))

Input: (tm '(mktrunk '(:ayona d-ahaba :alowaladu)))
(time taken 1.34 SECS)
Value: ((QP . :ayona) (VN . d-ahaba) (DN . :alowaladu))

Input: (tm '(mktrunk '(halo :anɔta darasota :alokitɛ|ba)))
(time taken 1.96 SECS)
Value: ((Q2 . halo) (DN . :anota) (VB . darasota) (DN . :alokita|ba))

Input: (tm '(mktrunk '(halo laka :aqola|mu+)))
(time taken 4.4e−1 SECS)
Value: ((Q2 . halo) ((PGP . laka) (GP . ka)) (NN . :aqola|mu+))

Input: (tm '(mktrunk '(s;addaqatoniy :alobinotu)))
(time taken 2.1 SECS) ("used in Table 63")
Value: (((VR . s;addaqatoniy) (CP . niy)) (DN . :alɔbinotu))

Input: (tm '(mktrunk '(:ibonu xa|lati samiyri+ rajulu+ qas;iyru+)))
(time taken 1.78 SECS)
Value: ((MD . :ibonu) (MD . xa|lati) (DN . samiyri+) (NN . rajulu+) (AA . qas;iyru+))

Input: (tm '(mktrunk '(la| walada svaqiyya d-ahaba :ilay :alomanozili)))
(time taken 2.12 SECS) ("used in Table 63")
Value: ((EP . la|) (VN . walada) (MD . svaqiyya) (VN . d-ahaba) (Pr . :ilay) (DN . :alomanozili))

Input: (tm '(mktrunk '(ɛinoda :alojabali :arobaɛatu :awola|di+ s;iga|ri+)))
(time taken 3.16 SECS)
Value: ((Pr . ɛinoda) (DN . :alojabali) (MM . :arobaɛatu) (NN . :awola|di+) (AA . s;iga|ri+))

Input: (tm '(mktrunk '(kataba samiyru+ sittata ɛasvara kita|ba+)))
(time taken 1.2 SECS)
Value: ((VB . kataba) (DN . samiyru+) (MM . sittata) (VM . ɛasvara) (NN . kita|ba+))

Input: (tm '(mktrunk '(qatala :aloqa|tilu wa|h-idata+ waɛisvoriyna binota+ jamiylata+)))
(time taken 4.52 SECS)
Value: ((VB . qatala) (DN . :aloqa|tilu) (MM . wa|h-idata+) (CMM . waɛisvoriyna) (NN . binota+) (AA . jamiylata+))

Input: (tm '(mktrunk '(:aloɛisvoruwna walada+ d;arabuw| ɛamora+)))
(time taken 2.54 SECS)
Value: ((DM . :aloɛisvoruwna) (NN . walada+) (VB . d;arabuw|) (DN . ɛamora+))

Input: (tm '(mktrunk '(sakanato :alobinotu fiy t-ala|t-ata ɛasvara manozila+)))
(time taken 2.84 SECS)

Value: ((VB . sakanato) (DN .. :alobinotu) (Pr . fiy) (MM . t-ala|t-ata) (VM . ɛasvara) (NN . manozila+))

Input: (tm '(mktrunk '(:innaniy ɨsvakarotu :alomuɛallima)))
(time taken 2.08 SECS)
Value: (((PC . :innaniy) (GP . nɨy)) (VB . svakarotu) (DN . :alomuɛallima))

Input: (tm '(mktrunk '(samiyru+ mamonuwh-u+ kita|ba+)))
(time taken 1.02 SECS)
Value: ((DN . samiyru+) (L2 . mamonuwh-u+) (NN . kita|ba+))

Input: (tm '(mktrunk '(ɛinoda :alojabali masa|kina s;agiyrata+)))
(time taken 2.56 SECS)
Value: ((Pr . ɛinoda) (DN . :alojabali) (XN . masa|kina) (AA . s;agiyrata+))

Input: (tm '(mktrunk '(ka|tibiy :alokita|bi :abona|:u zayodi+)))
(time taken 3.04 SECS)
Value: (((WN . ka|tibiy) (GP . iy)) (DN . :alokita|bi) (XN . :abona|:u) (DN . zayodi+))

Input: (tm '(mktrunk '(:arajulu+ fiy :alobayoti)))
(time taken 2.06 SECS)
Value: ((QNN . :arajulu+) (Pr . fiy) (DN . :alobayoti))

Input: (tm '(mktrunk '(:alaka :awola|du+)))
(time taken 1.88 SECS)
Value: (((QPGP . :alaka) (GP . ka)) (NN . :awola|du+))

Input: (tm '(mktrunk '(:ayona zayodu+)))
(time taken 2.0e−1 SECS)
Value: ((QP . :ayona) (DN . zayodu+))

Input: (tm '(mktrunk '(lamo yad-ohabo zayodu+ :ilay :alomadorasati)))
(time taken 2.66 SECS)
Value: ((JP . lamo) (VB . yad-ohabo) (DN . zayodu+) (Pr . :ilay) (DN . :alomadorasati))

Input: (tm '(mktrunk '(d-ahaba :ibonu ɛammi zayodi+ :ilay madorasati :alobana|ti)))
(time taken 3.6 SECS) ("**used in Table 63**")
Value: ((VN . d-ahaba) (MD . :ibonu) (MD . ɛammi) (DN . zayodi+) (Pr . :ilay) (MD . madorasati) (DN . :alobana|ti))

Input: (tm '(mktrunk '(mano qatala zayoda+ s;a|ra mad;oruwba+)))
(time taken 8.6e−1 SECS) ("**used in Table 63**")
Value: ((HQ . mano) (VB . qatala) (DN . zayoda+) (VB . s;a|ra) (L2 . mad;oruwba+))

Input: (tm '(mktrunk '(d-ahaba :alowaladu)))
(time taken 1.34 SECS)
Value: ((VN . d-ahaba) (DN . :alowaladu))

Input: (tm '(mktrunk '(ba|ta :alrrajulu ja|lisa+)))
(time taken 1.78 SECS)
Value: ((VB . ba|ta) (DN . :alrrajulu) (L1 . ja|lisa+))

Input: (tm '(mktrunk '(h-asibato :alo:ummu :alowaliyda qa|:ima+)))
(time taken 3.26 SECS)
Value: ((VB . h-asibato) (DN . :alo:ummu) (DN . :alowaliyda) (L1 . qa|:ima+))

Input: (tm '(mktrunk '(manah-a :alomudarrisu zakiyyata kita|ba+)))
(time taken 2.74 SECS)
Value: ((VB . manah-a) (DN . :alomudarrisu) (DN . zakiyyata) (NN . kita|ba+))

%%% **groups of words**

Input: (tm '(mktrunk '(katabotu darasota kasaroti farasvato qatalona| jalasotuma| nazala| farasvata| kunotumo svaribotunna h-asibuw| fatah-ona :as;iyru tukassiro taquwmiyna yaboniy tuganniy nad-ohaba tasokuna|ni yad/unna|ni tad/unna|ni takuwnuwna tasvorabona yah-osibuw| yafotah-ona :anotunna)))
(time taken 10.02 SECS) ("**used in Table 63**")
Value: ((VB . katabotu) (VB . darasota) (VB . kasaroti) (VB . farasvato) (VB . qatalona|) (VB . jalasotuma|) (VB . nazala|) (VB . farasvata|) (VB . kunotumo) (VB . svaribotunna) (VB . h-asibuw|) (VB . fatah-ona) (VB . :as;iyru) (VB . tukassiro) (VB . taquwmiyna) (VB . yaboniy) (VB . tuganniy) (VB . nad-ohaba) (VB . tasokuna|ni) (VB . yad/unna|ni) (VB . tad/unna|ni) (VB . takuwnuwna) (VB . tasvorabona) (VB . yah-osibuw|) (VB . yafotah-ona) (DN . :anotunna))

Input: (tm '(mktrunk '(takotubu taboniy taboniyona taboniyna tubonayona yasokuna|ni taboniya|ni yaboniya|ni jalasotu svaribona nad/unnu :alrrija|lu katabuw| :alowaliyda|tu katabona :alrrajulu :alomaro:ata qatalato darasona :alo:awwaluwna :alokabiyru :alokariymi :aloganiyyata :alnnisa|:u :aloqawiyma|tu :alsvsvara|:ifu :alt;t;awiylata|ni :alojabala|ni)))
(time taken 28.28 SECS) ("**used in Table 63**")
Value: ((VB . takotubu) (VB . taboniy) (VB . taboniyona) (VB . taboniyna) (VB . tubonayona) (VB . yasokuna|ni) (VB . taboniya|ni) (VB . yaboniya|ni) (VB . jalasotu) (VB . svaribona) (VB . nad/unnu) (DN . :alrrija|lu) (VB . katabuw|) (DN . :alowaliyda|tu) (VB . katabona) (DN . :alrrajulu) (DN . :alomaro:ata) (VB . qatalato) (VB . darasona) (DA . :alo:awwaluwna) (DA . :alokabiyru) (DA . :alokariymi) (DA . :aloganiyyata) (DN . :alnnisa|:u) (DA . :aloqawiyma|tu) (DA . :alsvsvara|:ifu) (DA . :alt;t;awiylata|ni) (DN . :alojabala|ni))

Input: (tm '(mktrunk '(:alowa|h-idu :alo:ah-adu :alrrajula :alowa|h-ida :alowa|h-idatu :alobinota :alowa|h-idata :alt-t-ala|t-u :alo:umma|ni walada|ni :alo:it-onatayoni :alowaliydayoni :alo:aqola|mu :alomi:atu :alofara|sva|ti :alo:alofa :alomada|risu :alo:a:immatu :aloxamosa :anota huwa hiya nah-onu :anotuma| huma| :anotumo humo hunna wa:aloɛisvoriyna :aloxamosata)))
(time taken 39.08 SECS) ("**used in Table 63**")
Value: ((DM . :alowa|h-idu) (DM . :alo:ah-adu) (DN . :alrrajula) (DM . :alowa|h-ida) (DM . :alowa|h-idatu) (DN . :alobinota) (DM . :alowa|h-idata) (DM . :alt-t-ala|t-u) (DN . :alo:umma|ni) (NN . walada|ni) (DM . :alo:it-onatayoni) (DN . :alowaliydayoni) (DN . :alo:aqola|mu) (DM . :alomi:atu) (DN . :alofara|sva|ti) (DM . :alo:alofa) (DN . :alomada|risu) (DN . :alo:a:immatu) (DM . :aloxamosa) (DN . :anota) (DN . huwa) (DN . hiya) (DN . nah-onu) (DN . :anotuma|) (DN . huma|) (DN . :anotumo) (DN . humo) (DN . hunna) (CDM . wa:aloɛisvoriyna) (DM . :aloxamosata))

B. AUTOMATIC TRACE FOR THE PARSE OF A V-COMPLEX:

':afasatad;oribuhunna', "and are you going to hit them?", tracing the functions: {exroot matchx derive1 derive2 derive3 mkperfect mkimperfect xparse futurize filter1 filter2 filter3 question filter6 mparse mktrunk}.

Input: (tm '(mktrunk '(:afasatad;oribuhunna)))

mktrunk with args:
  ARG1: (:afasatad;oribuhunna)
mparse called from mktrunk with args:
  ARG1: (:afasatad;oribuhunna)
filter6 called from wparse with args:
  ARG1: :afasatad;oribuhunna
question called from filter6 with args:
  ARG1: :afasatad;oribuhunna
futurize called from yparse with args:
  ARG1: :afasatad;oribuhunna
filter3 called from futurize with args:
  ARG1: :afasatad;oribuhunna
xparse called from filter3 with args:
  ARG1: :afasatad;oribuhunna
filter1 called from vparse with args:
  ARG1: :afasatad;oribuhunna
mkperfect called from filter1 with args:
  ARG1: :afasatad;oribuhunna
derive1 called from mkperfect with args:
  ARG1: :afasatad;oribuhunn
  ARG2: (patlist1 patlist2 patlist5)
exroot called from derive1 with args:
  ARG1: :afasatad;oribuhunn
  ARG2: (ls kn s;r bt bd; qm t;l bb jd xl s: sd bn t-n d-k m: gn svq :b :x ns d/nn jdd εmm :mm)
exroot= nil

%%% Henceforth—due to space constraints—suppressing calls to exroot which return nil.

derive1= nil
derive2 called from mkperfect with args:
  ARG1: :afasatad;oribuhunn
  ARG2: (patlist1 patlist2 patlist5)
derive2= nil
derive3 called from mkperfect with args:
  ARG1: :afasatad;oribuhunn
  ARG2: (patlist1 patlist2 patlist5)
derive3= nil

%%% Henceforth suppressing calls to derive1, 2, & 3 which return nil.

mkperfect= nil
filter1= nil
filter2 called from tvact1 with args:
  ARG1: :afasatad;oribuhunna
mkimperfect called from filter2 with args:

mkimperfect= nil
filter2= nil
filter1 called from vparse with args:
 ARG1: :afasatad;oribu
mkperfect called from filter1 with args:
 ARG1: :afasatad;oribu
mkperfect= nil
filter1= nil
filter2 called from tvact1 with args:
 ARG1: :afasatad;oribu
mkimperfect called from filter2 with args:
 ARG1: :afasatad;oribu
filter2= nil
filter1 called from vparse with args:
 ARG1: :afasatad;oribu|
mkperfect called from filter1 with args:
 ARG1: :afasatad;oribu|
mkperfect= nil
filter1= nil
filter2 called from tvact1 with args:
 ARG1: :afasatad;oribu|
mkimperfect called from filter2 with args:
 ARG1: :afasatad;oribu|
filter2= nil
xparse= nil
filter3= nil
futurize= nil
futurize called from yparse with args:
 ARG1: satad;oribuhunna
filter3 called from futurize with args:
 ARG1: satad;oribuhunna
xparse called from filter3 with args:
 ARG1: satad;oribuhunna
filter1 called from vparse with args:
 ARG1: satad;oribuhunna
mkperfect called from filter1 with args:
 ARG1: satad;oribuhunna
mkperfect= nil
filter1= nil
filter2 called from tvact1 with args:
 ARG1: satad;oribuhunna
mkimperfect called from filter2 with args:
 ARG1: satad;oribuhunna
mkimperfect= nil
filter2= nil
filter1 called from vparse with args:
 ARG1: satad;oribu
mkperfect called from filter1 with args:
 ARG1: satad;oribu
mkperfect= nil
filter1= nil
filter2 called from tvact1 with args:
 ARG1: satad;oribu
mkimperfect called from filter2 with args:
 ARG1: satad;oribu
mkimperfect= nil

filter2= nil
filter1 called from vparse with args:
 ARG1: satad;oribu|
mkperfect called from filter1 with args:
 ARG1: satad;oribu|
mkperfect= nil
filter1= nil
filter2 called from tvact1 with args:
 ARG1: satad;oribu|
mkimperfect called from filter2 with args:
 ARG1: satad;oribu|
mkimperfect= nil
filter2= nil
xparse= nil
filter3= nil
filter3 called from futurize with args:
 ARG1: tad;oribuhunna
xparse called from filter3 with args:
 ARG1: tad;oribuhunna
filter1 called from vparse with args:
 ARG1: tad;oribuhunna
mkperfect called from filter1 with args:
 ARG1: tad;oribuhunna
mkperfect= nil
filter1= nil
filter2 called from tvact1 with args:
 ARG1: tad;oribuhunna
mkimperfect called from filter2 with args:
 ARG1: tad;oribuhunna
filter2= nil
filter1 called from vparse with args:
 ARG1: tad;oribu
mkperfect called from filter1 with args:
 ARG1: tad;oribu
mkperfect= nil
filter1= nil
filter2 called from tvact1 with args:
 ARG1: tad;oribu
mkimperfect called from filter2 with args:
 ARG1: tad;oribu
derive1 called from mkimperfect with args:
 ARG1: ad;orib
 ARG2: (patlist3 patlist4 patlist6)
derive1= nil
derive2 called from mkimperfect with args:
 ARG1: ad;orib
 ARG2: (patlist3 patlist4 patlist6)
derive2= nil
derive3 called from mkimperfect with args:
 ARG1: ad;orib
 ARG2: (patlist3 patlist4 patlist6)
exroot called from derive3 with args:
 ARG1: ad;orib
 ARG2: (ld rd h-d sm :l h-sb h-sn krm kbr kt-r s;gr qs;r jml fqr t-mn svrf ktb svkr h-sb drs frsv qtl rbɛ skn fth- mnh- mnɛ d-hb xd;r jls ksr d;rb nzl h-ml qlm svbk ɛsvr sds sbɛ t-lt- t-mn tsɛ xms :lf s;fr h-dq fhm ɛlm qdm svrb rkb mr: sɛd rjl s;dq nfd- jbl h-mr zrq)

exroot= d;rb matchx called from derive3 with args:
ARG1: acocic
ARG2: ad;orib
matchx= t derive3= (patlist3 d;rb acocic)
mkimperfect= ((t ad;orib u) patlist3 d;rb acocic)
filter2= ((t ad;orib u) patlist3 d;rb acocic)
xparse= ((tad;oribu) (hunna))
filter3= ((tad;oribu) (hunna))
futurize= ((satad;oribu) (hunna))
question= ((:afasatad;oribu) (hunna))
filter6= ((:afasatad;oribu) (hunna))
mparse= (((:afasatad;oribu) (hunna)))

mktrunk= (((QCPVR . :afasatad;oribuhunna) (CP . hunna)))
(time taken 13.88 SECS)
Value: (((QCPVR . :afasatad;oribuhunna) (CP . hunna)))

End of LISP run after 36.88+5.34 SECS—79.9% store used;
xenakis %%% exit; script done on Thu Apr 26 11:44:51 1990.

==0==<*****>==0==

Appendix E
ARABIC-ENGLISH GLOSSARY OF LEXICAL ENTRIES

ENTRY	ENGLISH TRANSLATION
A. ENCLITIC PRONOUNS	
niy, iy, ya	"me, my, 1 (M, F) S, ACC-OBL".
na|	"us, our, 1 (M, F) D, P, ACC-OBL".
ka	"you, your, 2 (M) S, ACC-OBL".
ki	"you, your, 2 (F) S, ACC-OBL".
kuma|	"you, your, 2 (M, F) D, ACC-OBL".
kumo	"you, your, 2 (M) P, ACC-OBL".
kunna	"you, your, 2 (F) P, ACC-OBL".
hu, hi	"him, his, 3 (M) S, ACC-OBL".
ha|	"her, 3 (F) S, ACC-OBL".
huma|, hima|	"them, their, 3 (M, F) D, ACC-OBL".
humo, himo	"them, their, 3 (M) P, ACC-OBL".
hunna, hinna	"them, their, 3 (F) P, ACC-OBL".
B. SUBJECT PRONOUNS	
:ana|	"I, 1 (M, F) S, NOM".
nah-onu	"we, 1 (M, F) D, P, NOM".
:anota	"you, 2 (M) S, NOM".
:anoti	"you, 2 (F) S, NOM".
:anotuma|	"you, 2 (M, F) D, NOM".
:anotumo	"you, 2 (M) P, NOM".
:anotunna	"you, 2 (F) P, NOM".
huwa	"he, 3 (M) S, NOM".
hiya	"she, 3 (F) S, NOM".
huma|	"they, 3 (M, F) D, NOM".
humo	"they, 3 (M) P, NOM".
hunna	"they, 3 (F) P, NOM".
C. DEMONSTRATIVE PRONOUNS	
had-a|	"this (one), 3 (M) S, NEUT".
had-ihi	"this (one), 3 (F) S, NEUT".
had-a|ni	"these two, 3 (M) D, NOM".
had-ayoni	"these two, 3 (M) D, ACC-OBL".
ha|ta|ni	"these two, 3 (F) D, NOM".
ha|tayoni	"these two, 3 (F) D, ACC-OBL".
ha:ula|:i	"these (many), 3 (M, F) P, NEUT".

D. FREE PREPOSITIONS

fiy	"in, at".
mino	"from".
:ilay	"to, until".
ɛano	"on, about".
ɛalay	"on".
maɛa	"with".
ɛinoda	"with, at, near".
laday	"with".
fawoqa	"on, over".
baɛoda	"after, beyond".
tah-ota	"under, below".
:ama|ma	"in front of".
qabola	"before".
wara|:a	"behind".

E. BOUND PARTICLES

:a	Yes/no question operator.
sa	"going to".
li	"for, to, in order to".
la	"to".
bi	"at, in, with, for, through, by, for".
ka	"as, like".
wa	"and".
fa	"and, so, hence".
:alo	"the".

F. OTHER (FREE) PARTICLES

lamo	"not (with past reference)".
lano	"never (with future reference)".
h-attay	"in order to, so that, and".
kayo	"in order to, so that".
:inna	"indeed, verily, truly, it is true that".
:anna	"indeed, verily, truly, it is true that".
la|	"no, not (with present reference)".
ma|	"what".
sawofa	"will, shall".
qado	"perhaps, already".
halo	"whether (with a question)".
matay	"when".
:ayona	"where".
kayofa	"how".
ma|d-a|	"what".
mano	"who".
huna|	"here".
huna|ka	"there".
huna|lika	"over there".
:alo:a|na	"now".
:abada+	"never".
baɛodu	"yet".

G. ROOTS/STEMS	ENGLISH TRANSLATION	
	AS VERBAL	AS NOMINAL
ls	"not to be, not to exist".	N/A.
kn	"to be".	"being; universe, world".
kwwn	"to create, compose".	N/A.
s;r	"to become".	"becoming; future; destination".
s;yyr	"to transform into".	N/A.
bt	"to stay up the night".	"house".
byyt	"to ponder overnight".	N/A.
bd;	"(of the weather) to be hot; (of a twig) to dry up; (of a hen) to lay eggs; to rain; to be or become white".	"(being) white; egg; silver".
byyd;	"to improve, edit; to whiten".	N/A.
qm	"to stand up".	"standing up; people".
qwwm	"to make upright".	N/A.
t;l	"to be tall, long".	"(being) tall; height; length".
t;wwl	"to make taller, longer".	N/A.
bb	"to be a porter; to make an opening".	"opening, entry, door; chapter".
bwwb	"to divide into chapters, classify".	N/A.
jd	"to do well; to be or become good, generous".	"(being) good; goodness, generosity".
jwwd	"to improve; to recite (the Quran)".	N/A.
xl	"to manage, take care or charge of".	"(maternal) uncle or aunt".
xwwl	"to grant, allow".	N/A.
s:	"to make sb. suffer, to make them unhappy; to afflict, hurt, torment".	"(causing) suffering; evil; unhappiness, ugliness; (being) bad, evil".
sww:	"to scold, reproach, disapprove".	N/A.
sd	"to be or become master, lord, sovereign; to be honoured and glorified".	"(being) black; (being) master, lady; (being) honoured, glorified; nobility".
swwd	"to blacken; to make sb. master, chief".	N/A.
bn	"to build, establish".	"(being) child, son, daughter; birth; building, structure".
bnn	"to carry out intensive building activities".	N/A.
t-n	"to fold, double; to come second".	"(being) the second; two; Monday".
t-nn	"to increase to two; to divide into two".	N/A.
d-k	"to be intelligent".	"(being) intelligent; intelligence".
d-kk	"to grow old and fat".	N/A.
m:	"to make 100 fold, to join a group of 99 and make them 100".	"(being) the one hundredth; one hundred".
gn	"to be or become rich; to get married; to live".	"(being) rich; marriage, wealth; singing, cooing".
gnn	"to make richer; to sing, coo".	N/A.
svq	"to suffer, be unhappy".	"(being) unhappy; unhappiness, suffering, misery".
:b	"to be or become father".	"(being) father".
:x	"to befriend; to be or become brother or sister".	"(being) brother, sister, friend".
ns	"to neglect one's duty, one's work".	"neglect(ing) one's duty, one's work; women".

ld	"to give birth".	"birth, child; parent".
lld	"to assist in childbirth; to raise; bring up; to create out of".	N/A.
rd	"to flower; to arrive, come to, reach; to accrue".	"flower(ing); artery; imports; source".
rrd	"to blossom".	N/A.
h-d	"to be one, to unify".	"(being) one, unified, single; unit".
h-h-d	"to cause to be one, unified".	N/A.
sm	"to be or become handsome".	"(being) handsome; festival, season; decoration".
ssm	"to make more handsome; to confer distinction upon sb.".	N/A.
:l	"to come first, precede".	"(being) the first; precedence".
:wwl	"to explain, interpret".	N/A.
d/nn	"to think, consider, believe".	"doubt(ing), uncertainty; (being) suspect; site".
jdd	"to cut; to strive; to be or become new".	"cutting; (being) serious; new; grandfather, grandmother".
jddd	"to renew".	N/A.
ɛmm	"to spread; to be a (paternal) uncle or aunt".	"(being) a (paternal) uncle or aunt; (being) general, public; turban".
ɛmmm	"to cause to spread; to generalize; to put on a turban".	N/A.
h-sb	"to think, deem; to count".	"calculation, arithmetic; esteem, dignity, honour; punishing".
h-sn	"to be or become good, beautiful".	"(being) good, beautiful; beauty".
h-ssn	"to improve; to dress up, to make up; to decorate".	N/A.
krm	"to be or become generous, kind".	"(being) generous, kind; generosity, kindness".
krrm	"to honour, dignify, glorify".	N/A.
kbr	"to be or become large, big; to become senior".	"(being) big, large, senior; size; age".
kbbr	"to make bigger, larger".	N/A.
kt-r	"to abound, to be thick on the ground".	"(being) abundant, plentiful, frequent; multitude".
kt-t-r	"to cause to be spread, plentiful, frequent".	N/A.
s;gr	"to be or become small, little".	"(having a) small size; youth".
s;ggr	"to make smaller, decrease".	N/A.
qs;r	"to be or become short".	"(being) short; palace".
qs;s;r	"to shorten".	N/A.
jml	"to improve in beauty and manners".	"(being) beautiful; beauty; camel".
jmml	"to make more beautiful".	N/A.
fqr	"to be or become poor".	"(being) poor; poverty".
fqqr	"to make holes into".	N/A.
t-mn	"to be or become expensive; to make eightfold, to join a group of seven and make them eight".	"(being) the eighth; eight; price".
t-mmn	"to price, cost; to divide into eight".	N/A.
svrf	"to be or become honourable, dignified, glorified".	"honour, dignity, glory, nobility; (being) honourable, dignified, glorified".
svrrf	"to honour, dignify, glorify".	N/A.
ktb	"to write".	"writing; written material, book".

kttb	"to teach or cause to write".	N/A.
svkr	"to thank, praise".	"praising; praise, thanks, gratitude".
drs	"to study".	"study(ing); lesson; school".
drrs	"to cause to study, to teach".	N/A.
frsv	"to lay out or spread (a sheet, rug, or bedding)".	"laying out; bed; butterfly".
frrsv	"to lay out a bed for sb.".	N/A.
qtl	"to kill".	"killing; murder".
qttl	"to massacre".	N/A.
rbɛ	"to make fourfold, to join a group of three and make them four; to take the fourth of".	"(being) the fourth; four; garden".
rbbɛ	"to make something square".	N/A.
skn	"to dwell; to be or become still, peaceful".	"dwelling, house; peace, quiet; stillness".
skkn	"to cause to be still, mute, peaceful, quiet".	N/A.
fth-	"to open".	"opening; key".
ftth-	"to open (all doors)".	N/A.
mnh-	"to grant; to give".	"giving; grant; recipient".
mnɛ	"to prevent, deprive; to make (a Nominal) indeclinable".	"prohibition, deprivation; obstacle; indeclinability".
d-hb	"to go to, head towards".	"going; departure; direction; gold; school, ideology".
xd;r	"to be or become green".	"(being) green".
xd;d;r	"to make green".	N/A.
jls	"to sit down".	"sitting; seat; council; meeting; companion".
jlls	"to cause to sit down".	N/A.
ksr	"to break".	"break(ing); fracture(d)".
kssr	"to shatter".	N/A.
d;rb	"to hit (the target), to hit sb.; to give (an example); to pitch (a tent); to say (a prayer)".	"hit(ting); type, example; tax; place or time of hitting".
d;rrb	"to beat up (severely)".	N/A.
nzl	"to go down, travel, arrive".	"travelling, arrival; house, hotel; guest; rank, position".
nzzl	"to cause to go down, reach; to host; to inspire with (the Quran)".	N/A.
h-ml	"to carry; to betray; to become pregnant".	"weight, burden; pregnancy; (being) pregnant; embryo".
h-mml	"to cause to carry".	N/A.
qlm	"to cut, clip".	"cutting; pen; container, case; receptacle".
qllm	"to pare".	N/A.
svbk	"to entwine".	"entwining; net, trap, snare; window".
svbbk	"to interweave, entangle".	N/A.
ɛsvr	"to befriend; to make tenfold, to join a group of nine and make them ten".	"(being) the tenth; ten; tribe; (being a) friend; friendship".
ɛsvsvr	"to increase to ten; to take the tenth of".	N/A.
sds	"to make sixfold, to join a group of five and make them six".	"(being) the sixth; six".
sdds	"to divide into six".	N/A.
stt	N/A.	"six".

sbɛ	"to make sevenfold, to join a group of six and make them seven; to divide into seven; to take the seventh of".	"(being) the seventh, seven; lion; the place where lions abound".
sbbɛ	"to increase to seven".	N/A.
t-lt-	"to make threefold, to join a group of two and make them three; to be third; to divide into three".	"(being) the third; three".
t-llt-	"to triple".	N/A.
tsɛ	"to make ninefold, to join a group of eight and make them nine; to take the ninth of".	"(being) the ninth; nine".
xms	"to make fivefold, to join a group of four and make them five".	"(being) the fifth; five".
xmms	"to divide into five".	N/A.
:lf	"to give sb. 1000 dollars, pounds, etc.".	"(giving sb.) a 1000 dollars, pounds, etc.; friendship; (being) a friend; writer".
:llf	"to complete the 1000; to make up, create".	N/A.
s;fr	"to be or become yellow; to whistle".	"whistling; (being) yellow".
s;ffr	"to make yellow".	N/A.
h-dq	"to look at; to hit sb. in the eye; to surround".	"looking; surrounding; eye; garden".
h-ddq	"to stare, gaze".	N/A.
fhm	"to understand, know".	"understanding, knowledge".
fhhm	"to cause to understand, to explain".	N/A.
ɛlm	"to know, realize".	"knowing, knowledge, science; flag; (being) knowledgeable".
ɛllm	"to cause to know, learn".	N/A.
qdm	"to come, arrive; to go to".	"arriving, arrival; foot".
svrb	"to drink".	"drinking (place); drink, water; (drinking) receptacle".
svrrb	"to cause to drink; to drench; to imbue".	N/A.
rkb	"to mount, ride; to brave (dangers); to travel (by sea, road, etc.)".	"ship, boat, ferry, carriage, cab, vehicle; camel; travelling, mounting; travelling companion".
rkkb	"to assemble; to cause to mount, ride".	N/A.
mr:	"to be effeminate".	"(being) effeminate; woman; person; (being) generous, chivalrous".
mrr:	"to wish sb. good health from their drink or food".	N/A.
sɛd	"to be happy, lucky".	"(being) happy, lucky; forearm; happiness; luck, good fortune".
rjl	"to be on foot".	"(being) on foot; man; comb; pedestrian".
rjjl	"to make stronger; to comb (one's hair)".	N/A.
s;dq	"to be honest, truthful; to befriend".	"(telling) the truth; honesty; friendship; friend".
s;ddq	"to believe, trust".	N/A.
nfd-	"to penetrate; to escape; to leave behind".	"window, opening, escape; penetration".
nffd-	"to execute; to cause to penetrate; to send".	N/A.

jbl	"to create, to mould; to force or coerce".	"creation; mountain; crowd (of people), tribe; (of a man (being)) ugly".
jbbl	"to shred to pieces".	N/A.
h-mr	"to shave (one's head); to peel".	"shaving; (being) red; donkey; redness; (being) vile, ignoble".
h-mmr	"to redden; to call sb. a donkey".	N/A.
zrq	"to be or become blue; (of eyes) to turn over and become white".	"(of eyes) turning white; the sky; (being) blue; blueness".
:mm	"to be or become a mother; to lead the way".	"(being) a mother; (prayer) leadership; (being) a prayer leader; (being) a source, nation, community, people; (being) of handsome figure; motherhood, maternity".
:mmm	"to nationalize".	N/A.

==0==<★★★★★>==0==

Appendix F

ARABIC-ENGLISH GRAMMAR TERMINOLOGY

Abbreviated Nominal ..... اسم مقصور

ablaut ..... مغايرة

abstract ..... معنوي

accusative ..... حالة النّصب

active ..... مبنيّ للمعلوم

activization ..... البناء للمعلوم

Adjective ..... نعت

Affirmative Particle ..... أداة توكيد

affix ..... زائدة

affixation ..... زيادة

agglutinative language ..... لغة لصقيّة

agreement ..... تطابق / توافق

':alif' of Abbreviation ..... الألف المقصورة

':alif' of Prolongation ..... الألف الممدودة

ambiguity ..... لبس، التباس

annexation/genitive construction ..... الإضافة / تركيب إضافيّ

Annexed ..... المضاف

antefix ..... ما قبل السّوابق (من زوائد)

Aplastic Verbs ..... الأفعال الجامدة

argument ..... مسند إليه

aspect ..... الجهة

assertion/emphasis ..... تأكيد

Assertive Particle ..... أداة إثبات

Assimilated Adjective ..... صفة مشبّهة

assimilation ..... الإدغام/المماثلة

Attributive ..... وصفي

augmentation ..... صياغة المزيد

Augmentative infixes ..... حروف زيادة داخلية

Augmented ..... المزيد

Auxiliary ..... فعل مساعد

Basic ..... المجرّد

Bi-Augmented ..... مزيد بحرفين

bound Accusative Pronoun ..... ضمير نصب متّصل

Bound Particle ..... أداة متّصلة

Bound Preposition ..... حرف جرّ متّصل

Broken Plural ..... جمع تكسير

CASE ..... إعراب الاسم

CASE MARK ..... علامة إعراب الاسم

categorization ..... تقسيم

CATEGORY ..... قسم

causativization ..... صياغة الفعل للسّببيّة

clitic ..... لاصق

Collective Noun ..... اسم جمع

collocutive ..... الإشارة للحاضر أو المخاطب

Common Noun ..... اسم شائع

Complement ..... متعلّق

compound ..... مركّب

Conditional Particle ..... أداة الشّرط

Conditional Pronoun ..... اسم الشّرط

Conjugable/Plastic Verbs ..... الأفعال المتصرّفة

conjugation ..... تصريف الفعل

consonant ..... حرف

contingency ..... تعليق

conventional ..... اتّفاقًا

coordination ..... العطف

Coordinative/Coordinate Conjunction ..... حرف العطف

Countable Noun ..... اسم قابل للعدّ

declarative ..... إخباريّ

declension ..... المنع من الصّرف وعدمه

declinability ..... تمكّن

declinable Nominal ..... اسم متمكّن/معرب/منصرف

deep CASE ..... محلّ الإعراب

Defective Nominal ..... اسم غير صحيح

Defective Verb ..... فعل معتلّ

definite ..... معرفة

DEFINITENESS ..... التّعريف والتّنكير

definitization ..... تعريف

definitizer ..... معرّف

Demonstrative Pronoun ..... اسم الإشارة

dependent ..... تابع

Dependent ..... مضاف إليه

derivation ..... اشتقاق

Derivative ..... اشتقاقيّ

Derived ..... مشتقّ

Derived Substantive ..... اسم ذات مشتقّ

Determiner ..... أداة التّعريف

Diacritics ..... الشّكل ، التحريك

Dimunitive Noun ..... تصغير

Diptote ..... اسم ممنوع من الصّرف / متمكّن غير أمكن

Diptote of Type 1 ..... ممنوع من الكسر

Diptote of Type 2 ..... ممنوع من النّصب

Diptote of Type 3 ..... ممنوع من الضّم

Discontinuous Double Defective ..... اللّفيف المفروق

ditransitive Verb ..... فعل متعدّ إلى مفعلين

Doubled Triliteral ..... مضعّف ثلاثيّ

doubling/reduplication ..... تضعيف

dual ..... مثنّى

duality ..... ثنائيّة

dualization ..... تثنية

elision ..... الحذف

enclitic ..... لاحق

enclitic Pronoun ..... ضمير لاحق متّصل

Exceptive Particle ..... أداة استثناء

exlocutive ..... (الإشارة) للغائب

expression ..... عبارة / شبه جملة

Extended Nominal ..... اسم ممدود

Factitive Verbs ..... أفعال التّصيير

feminine ..... مؤنّث

feminization ..... تأنيث

FLEXION ..... التّنوين وحذف النّون من عدمهما

Free Accusative Pronouns ..... ضمائر النّصب المنفصلة

Free Particle ..... أداة منفصلة

Free Preposition ..... حرف جرّ منفصل

freezing ..... بناء

frozen ..... جامد

fully declinable Nominal ..... اسم متمكّن أمكن

Future Particle ..... أداة مستقبلية

futurization ..... صياغة الفعل للمستقبل

gemination ..... شدّة

GENDER ..... التذكير والتأنيث

Generic Noun ..... اسم جنس

Genitive Pronouns ..... ضمائر الجرّ

genitivization ..... صياغة الاسم في حالة الجرّ

GOVERNMENT ..... عمل

governor/regent ..... عامل

grapheme ..... حرف مجرّد

graphemic ..... متعلّق بالكتابة بالنظر إلى المستوى المجرّد

graphological ..... من باب دراسة الخطّ

graphotactic ..... مقيّد بقواعد كتابة الخطّ

head ..... متبوع

homonymy ..... اشتراك / تجانس لفظيّ

Hollow Verb ..... فعل أجوف

human ..... العاقل

Hyper Plural ..... منتهى الجموع

idiosyncratic patterning ..... أوزان غير قياسية

imperative ..... الأمر

imperfect ..... المضارع

Inchoative Verbs ..... أفعال الشروع

indeclinable Nominal ..... اسم غير متمكّن / مبنيّ

indefinite ..... نكرة

indicative ..... المضارع المرفوع

Indicator ..... عامل المضارعة

Infinitive ..... المصدر

infix ..... داخلة

infixation ..... إدخال

inflection ..... تصريف/إعراب الاسم أو الفعل

Inflector ..... جار

insertion ..... زيادة

intensification ..... صياغة الفعل للتوكيد

Interrogative Particle ..... أداة الاستفهام

intransitive Verbs ..... الأفعال اللازمة

jussive ..... المضارع المجزوم

Jussive Particle ..... أداة جزم الفعل

Jussor ..... جازم

lexical ..... لفظي

liaison ..... الوصل

Lunar characters ..... الحروف القمرية

masculine ..... مذكر

metathesis ..... القلب

M-Infinitive ..... المصدر الميمي

modality ..... مساعدة الفعل

mode ..... صياغة الجملة للاستفهام أو عدمه

Mono-Augmented ..... مزيد بحرف واحد

monotransitive Verb ..... فعل متعدٍ إلى مفعول واحد

MOOD ..... إعراب الفعل

MOOD MARK ..... علامة إعراب الفعل

morphological ..... صرفي

morphology ..... علم الصرف

mutation ..... إعلال

mute/quiescent ..... ساكن

negation ..... نفي

Negative Particle ..... أداة نفي

neutral ..... محايد

neutralization ..... تحييد

Nominal ..... اسم

nominal group/expression ..... عبارة اسمية

nominal sentence ..... جملة اسمية

nominative ..... حالة الرفع

non-human ..... غير العاقل

non-regent ..... غير عامل

Noun ..... اسم ذاتة أو معنى

Noun of Comparison/Comparative ..... اسم التفضيل

Noun of Exaggeration/Exaggerative ..... اسم المبالغة

Noun of Instance ..... اسم المرة

Noun of Instrument/Instrumental ..... اسم الآلة

Noun of Manner ..... اسم النوع/الهيئة

Noun of Origin ..... نسبة

Noun of Place/Noun of Time/Tlocative ..... اسم زمان/اسم مكان (زمكان؟)

Noun of the Infinitive ..... اسم المصدر

Noun of the Verb ..... اسم الفعل

NUMBER ..... العدد

Numeral ..... اسم العدد

Oath Particle ..... حرف القسم

Object ..... مفعول به

oblique prepositional/genitive ..... حالة الجر

Particle ..... أداة/حرف

passive ..... المبني للمجهول

passivization ..... البناء للمجهول

pattern ..... الوزن/الميزان

perfect ..... الماضي

PERSON ..... الشخص

phonetic ..... صوتي، فونيتيكي

phonotactic ..... مقيد بقواعد الصوتيات

Place Adverb ..... ظرف مكان

plural ..... جمع

plurality ..... دلالة الاسم على الجمع من عدمه

pluralization ..... صياغة الجمع

Plural of Abundance ..... جمع الكثرة

Plural of Paucity ..... جمع القلة

Poly-Augmented ..... مزيد بأكثر من حرف

Polysynthetic/Polymorphemic Word ..... كلمة مركبة من كلمة أساسية وعدة زوائد

precedence ..... تقدم

predicate ..... مسند

predicative ..... إسنادي

prefix ..... سابقة

prefixation ..... إسباق

Preposition ..... حرف جر

Primitive ..... اسم غير مشتق

priority ..... صدارة

proclitic ..... سابقة

Prohibitive Particle ..... أداة نهي

Prolongation ..... مد

Pronoun ..... ضمير

Proper Noun ..... الاسم العلم

Pseudo-Subject ..... نائب الفاعل

Quadriliteral ..... رباعيّ

Qualificative ..... صفة

Quasi-Sound Verb ..... فعل مثال

question ..... استفهام

radical ..... أصليّ

rationalization ..... عقليّ

REFERENCE ..... لإشارة عائدة

regimen ..... معمول

regularization ..... اتّباع القياس

regular patterning ..... أوزان قياسيّة

Regular Plural ..... جمع سالم

Relative Pronoun ..... الاسم الموصول

root ..... جذر/أصل الفعل

rule ..... قاعدة

semantic ..... دلاليّ

semivowel/weak letter ..... حرف العلّة

sentence ..... جملة

Sequential Double Defective ..... المضعّف المقرون

sexual ..... جنسًا

singular ..... مفرد

soft vowels ..... حروف اللّين

Solar characters ..... الحروف الشمسيّة

Solid Verb ..... فعل سالم

Sound Nominal ..... اسم صحيح

Sound Verb ..... فعل صحيح

specification construction ..... تركيب تمييز

specifier ..... تمييز

state ..... حالة

stem ..... جذر الاسم

Subject ..... الفاعل

Subject Pronouns ..... ضمائر الرفع

subjunctive ..... المضارع المنصوب

Subjunctive Particle ..... أداة نصب للفعل

Subjunctor ..... ناصب للفعل

subordination ..... إتباع

Substantive Nominal ..... اسم ذات

substitution ..... إبدال

suffix ..... لاحقة

suffixation ..... إلحاق

'sukuwn'/quiescence ..... سكون

Super Plural ..... جمع الجمع

Supernumerary character ..... حرف زيادة

surface CASE ..... الإعراب الظاهر

syntactic ..... نحوي / تركيبي

syntax ..... علم النحو

temporary indeclinability ..... بناء عرضي

TENSE ..... الزمن

Time Adverb ..... ظرف زمان

transitive Verbs ..... الأفعال المتعدية

TRANSITIVITY ..... التعدية

Transmutative Verbs ..... أفعال التحويل

tree/marker ..... مُنشجر

Tri-Augmented ..... مزيد بثلاثة أحرف

Triliteral ..... ثلاثي

true feminine ..... المؤنث الحقيقي

type ..... نوع / صنف

underlying ..... مقدّر

Verb ..... فعل

Verbal ..... قائم بوظيفة الفعل

Verbal1/Active Verbal/Noun of the Agent ..... اسم فاعل

Verbal2/Passive Verbal/Noun of the Patient ..... اسم مفعول

verbal expression ..... عبارة فعلية

verbal sentence ..... جملة فعلية

Verbs of Approximation ..... أفعال المقاربة

Verbs of Certainty ..... أفعال اليقين

Verbs of Doubt ..... أفعال الشك

Verbs of Praise ..... أفعال المدح

Verbs of Probability ..... أفعال الرجحان

Verbs of Surprise ..... أفعال التعجب

Verbs of the Heart ..... أفعال القلب

Verbs of Vituperation ..... أفعال الذم

vocalized ..... متحرك

Vocative Particle ..... أداة نداء

VOICE ..... بناء الفعل للمعلوم أو المجهول

vowel ..... حركة

vowels of Prolongation ..... حروف المدّ

Weak Nominal ..... اسم منقوص

Weak Verb ..... فعل ناقص

==0==<*****>==0==